Selective Encryption Techniques for Secure Transmission of MPEG Video Bit-Streams

by

Ghassan Ibrahim Al-Regib

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

December, 1998

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Selective Encryption Techniques for Secure Transmission of MPEG Video Bit-Streams

BY

GHASSAN IBRAHIM AL-REGIB

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

Electrical Engineering

December, 1998

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

This thesis, written by

GHASSAN IBRAHIM AL-REGIB

Under the direction of his thesis committee, and approved by all the members, has been presented to and accepted by the dean, College of Graduate Studies, in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

Thesis Committee:

Chairman (Dr. S. A. Al-Semari)

Co-Chairman (Dr. A. M. Alattar)

Member (Dr. M. M. Dawoud)

Member (Dr. U. A. Al-Suwailem)

Department Chairman

Dean, College of Graduate Studies

Date: 2-2-1999

To my mother

and

In memory of my father

Acknowledgments

First and for most my deep appreciation goes to my thesis advisors Dr. Saud Al-Semari and Dr. Adnan Alattar for their constant help, guidance and great amount of precious time their devoted to my academic development.

I would like to thank my thesis committee members Dr. M. M. Dawoud and Dr. U. A. Al-Suwailem for their interest, cooperation, advice and constructive criticism.

My heartfelt thanks and gratefulness also goes to my mother for her moral support and encouragement.

Finally, I wish to thank my brothers Hisham, Hussam and Emad for the financial support and the continuous encouragement they provide me with during my M.S. study.

Contents

Acknowledgments iv

Contents v

List of Figures viii

List of Tables xii

Abstract xvii

Abstract (Arabic) xviii

Chapter 1 Introduction 1

1.1 General 1

1.2 Scrambling 3

1.3 Brief Review on Cryptology 6

1.4 Mathematical Background 16

1.5 Popular Cryptographic Algorithms 20

1.6 MPEG Video Compression Standard 32

1.7 Statement of the Problem 41

Chapter 2 Encrypting the I-Frames of MPEG- Compressed Video Bit-streams-An Evaluation 43

2.1 Introduction 43

2.2 Research trends in securing digital compressed video 45

2.3 Thesis experiment set-up 50

2.4 Encrypting I-frames 54

2.5 Processing time of the suggested improvements on the method of encrypting I-frames 64

2.6 Summary 69

Chapter 3 Improving the Security Level of Selectively-Encrypted MPEG-I Video Bit-streams 71

3.1 Introduction 71

3.2 The first proposed classification 72

3.3 Encrypting all I-macroblocks in all frames 73

3.4 The second proposed classification 96

3.5 Further encrypting the headers of P- and B-macroblocks 96

3.6 Alternative encryption/decryption 106

3.7 Encrypting 64-bit segment from every other macroblock 114

3.8 Summary 118

Chapter 4 Analyzing the Security Level of the Selective Encryption Methods 121

4.1 Introduction 121

4.2 Attacking the Proposed Schemes Without Crypt-analysis 123

4.3 Transmitting the DES Key within MPEG-I video bit stream 144

4.4 Cryptanalyst's Attacks on DES and RSA Keys 146

4.5 Summary 159

Chapter 5 Conclusions and Future Research 161

5.1 Conclusions and summary 161

5.2 Future Research 165

Appendix 167

Nomenclature 169

Bibliography 170

List of Figures

Figure 1.1: The active line rotation principle..... 3

Figure 1.2: The Modified Active Line Rotation Principle..... 4

Figure 1.3: Branches of cryptology..... 7

Figure 1.4: A PRSR with polynomial $(1+X^2+X^3)$..... 10

Figure 1.5: Private-Key (Symmetric) Ccrypto-system..... 12

Figure 1.6: Public-Key (Asymmetric) Crypto-system..... 12

Figure 1.7: A typical secure video system..... 20

Figure 1.8: The Standard Building Block (SBB)..... 23

Figure 1.9: The DES-Key shuffling process..... 24

Figure 1.10: The DES algorithm..... 25

Figure 1.11: The main layers of an MPEG-I bit-stream..... 34

Figure 1.12: Layers of MPEG video bit-stream by looking at them through: (a) the displayed video clip and (b) the bit-stream. (GOP indicates a group of pictures; MB indicates a macroblock.)..... 37

Figure 1.13: A pictorial representation of predicting the P- and B- frames..... 39

Figure 2.1: Structure of MPEG-I Bit-stream Parser..... 52

Figure 2.2: The first nine pictures of the I-frames-encrypted *"Para"* video clip. The numbers represent the temporal reference of each picture..... 57

Figure 2.3: Selected frames from the I-frames-encrypted mode of *"Para"* video clip..... 58

Figure 2.4: Two consecutive frames from the encrypted (or zeroed) *"Gulf-War"* video clip at a scene change; (a) is before the scene change and (b) after the scene change..... 58

Figure 2.5: The zeroed I-frame with the MB that contains a random pattern of colors. (a) is the upper left corner of the image in (b) with the MB that has a random color pattern. (b) is the first encrypted (zeroed) I-frame..... 59

Figure 2.6: Sample frames of the I- and P-frames encrypted *"Gulf-War"* video clip..... 61

Figure 2.7: A plot of τ versus F for two video clips. "— and -" are for the *"Fibon"* and the *"Fish"* video clips, respectively..... 68

Figure 3.1: Illustration the process of zeroing the I-MB's. (a) The original slice containing two I-MB's only. (b) The slice after the parser replaces the required code words to implement zeroed I-MB's..... 79

Figure 3.2: Selective Encryption of the *"Para"* video clip. (a) after zeroing all I-frames only. (b) after zeroing all I-MB's in all frames. The numbers are the frame number in the video clip..... 82

Figure 3.3: Selective Encryption of the "*Gul-War*" video clip after decoding using standard MPEG-I decoder. (a) and (c) after zeroing I-frames only (b) and (d) after zeroing I-MB's in all frames..... 83

Figure 3.4: A Frame of the *"Para"* video clip resulted by decoding the real encrypted sequence with the VMPEG 1.7d-Lite decoder..... 84

Figure 3.5: Frames from *"Fibon"* video clip. (a) Two frames from the I-MB's-encrypted video decoded by VMPEG 1.7d-Lite. (b) Two frames from the original video clip for comparison..... 85

Figure 3.6: The start and the end limits of the encrypted portion of the I-MB in the method of encrypting all I-MB's in all frames..... 86

Figure 3.7: A block diagram showing the process the receiver has to perform, in case the I-MB's are encrypted, to determine the end of the encrypted portion of

the I-MB..... 92

Figure 3.8: An example illustrating two schemes to encrypt the headers of the non-intra macroblocks within a slice..... 99

Figure 3.9: Encrypting P- and B-macroblocks' headers in addition to all I-MB's. These frames are obtained by decoding the encrypted video by the VMPEG 1.7d-Lite decoder..... 102

Figure 3.10: The process of differentially encodeing the DC value in the blocks of I-MB's. (Similar figure exists in [42] as Figure 5.5)..... 116

Figure 4.1: Histograms for the '*Fibon*' Motion Vectors (MV). (a) **Forward MV_x.** (b) **Forward MV_y.** (c) **Backward MV_x** and (d) **Backward MV_y.** (The X-axis represents the length of the motion vector.)..... 129

Figure 4.2: MPEG-1 Block Decoding Loop..... 138

Figure 4.3: The effect of shifting down the highest two DCT-coefficients of the blocks in a macroblock..... 144

Figure 4.4: The syntax in the Picture header to search for any user data sent within MPEG-I video bit stream..... 145

List of Tables

Table 1.1: Time and cost needed to identify the key by Brute-Force attack for a machine that can test 1-G keys/sec..... 15

Table 1.2: Time and cost needed to identify the key by Brute-Force attack for a machine that can test *L* keys/sec; *c* is the cost of one machine..... 16

Table 1.3: Processing time for encrypting video frames using DES..... 27

Table 1.4: Processing time for encrypting video frames using RSA..... 30

Table 1.5: Comparison between the DES and the RSA algorithms..... 31

Table 1.6: Functions and start codes of the layers in an MPEG-I compressed video sequence 35

Table 2.1: Encoder parameters..... 54

Table 2.2: Statistics on the test video clips for the data content in the I-frames..... 62

Table 2.3: Average number of bits in both one I-frame and one non-intra-frame..... 63

Table 2.4: τ obtained by encrypting I-frames only..... 63

Table 2.5: The fraction of the video data being encrypted, τ , for different I-frame frequencies for the *"Fibon"* video clip..... 66

Table 2.6: The fraction of the video data being encrypted, τ , for two I-frame frequencies (F) for seven video clips..... 67

Table 2.7: The fraction of the video data being encrypted, τ , resulting from encrypting all I- and P-frames..... 69

Table 3.1: Variable Length Codes (VLC's) for the macroblock_type code word that exists in the macroblock header for (a) P-Pictures (b) B-Pictures. (c) Gives the interpretation of the flags (I, B, F, Q) listed in tables (a and b)..... 75

Table 3.2: Variable Length Codes (VLC) for (a) the DCT_DC_SIZE_luminance code word and (b) the DCT_DC_SIZE_chrominance code word..... 76

Table 3.3: Statistics on the test video clips for the data content of the I-MB's..... 87

Table 3.4: Percentage of the data contained by I-MB's..... 88

Table 3.5: The fraction of video data being encrypted, τ , for the two methods of selective encryption: encrypting only I-frames and encrypting all I-MB's in all frames..... 88

Table 3.6: Statistics on the number of bits in those I-MB's that are in P- and B-frames..... 90

Table 3.7: A list of the lengths of the valid MB-Data code words with the number of those code words that have the corresponding length..... 93

Table 3.8: The average delay caused by the process illustrated in Figure (3.7) for the test video clips used in this thesis..... 95

Table 3.9: The average delay caused by the process illustrated in Figure (3.7) assuming t_{VLC} is 3.33 *nsec* for the test video clips used in this thesis..... 95

Table 3.10: The fraction of video data being encrypted, τ, for the methods of encrypting all I-MB's and the method of further encrypting the headers of non-intra MB's..... 103

Table 3.11: Comparison of τ between the three methods of selective encryption discussed so far..... 104

Table 3.12: The inefficiency, in the method of encrypting all I-MB's as well as the headers of the non-intra macroblocks, introduced by encrypting part of the data of the non-intra macroblocks..... 106

Table 3.13: Statistics showing the τ for both methods of encrypting all I-MB's and half of them..... 108

Table 3.14: The fraction of the video data being encrypted, τ, for both the method discussed in section 3.5 and the method of encrypting half I-MB's and all the headers of the non-intra macroblocks..... 110

Table 3.15: The inefficiency, in the method of encrypting half the I-MB's as well as the headers of all the non-intra macroblocks, introduced by encrypting part of the data of the non-intra macroblocks..... 111

Table 3.16: The fraction of the video data being encrypted, τ, for three methods..... 112

Table 3.17: The inefficiency, in the method of encrypting half the I-MB's as well as the headers of half the non-intra macroblocks, introduced by encrypting part of the data of the non-intra macroblocks..... 113

Table 3.18: The fraction of the video data being encrypted, τ, for the method of encrypting 64-bit segment from every other macroblock..... 118

Table 4.1: The syntax of the macroblock header in MPEG-I bit stream..... 126

Table 4.2: The length (in bits) and the number of possible values for each parameter in the macroblock header. (* The number of possible values for the **macroblock_type** parameter depends on the picture type.)..... 127

Table 4.3: The average number of times to decode one macroblock in a P- or a B-picture with a certain **macroblock_type** parameter..... 132

Table 4.4: The maximum number of frames at which the same key can be used to encrypt all frames without having a threat from differential cryptanalysts. (Assuming all frames are of type intra.)..... 148

Table 4.5: Time and cost needed to identify the DES key by Brute-Force attack for a machine that can test L keys/sec; c is the cost of one machine..... 150

Table 4.6: Time and cost needed to identify the DES key by Brute-Force attack for a machine that can test 76460 keys/sec; the cost of one machine is U.S. $500..... 151

Table 4.7: Time and cost needed to identify the DES key by Brute-Force attack for two different state-of-the-art machines..... 152

Table 4.8: Historical records of the successful large integer factorizations..... 155

Table 4.9: The MIPS-years requirements to factor different RSA-modulus lengths as estimated by Odlyzko [40]..... 156

Table 4.10: The estimated time and cost required for factoring an RSA-modulus with different lengths and in different years. (The cost of one machine is U.S. $500)..... 158

Abstract

Name: **Ghassan Ibrahim Al-Regib**

Title: **Selective Encryption Techniques for Secure Transmission of MPEG Video Bit-Streams**

Major Field: **Electrical Engineering**

Date of Degree **December 1998**

Thorough encryption/decryption of MPEG video bit-stream requires large processing time. However, through selective encryption of parts of the bit-stream, this processing time can be highly reduced while a reasonable level of security is maintained. Some researchers proposed encrypting only the data associated with the I-frames only. They suggested that the P and B frames would inherit their security from the security of the encrypted I-frames. In this thesis, some experimental results are presented to show that this method of encryption does not provide adequate level of security. Next, four better methods of encryption that provide more security at low processing time are proposed in this thesis. These methods are: encrypting the Intra-coded macroblocks in all frames (i.e. I-, P- and B-frames) of the MPEG video sequence, encrypting in addition to the I-macroblocks the headers of the non-intra macroblocks, encrypting alternatively the Intra-coded macroblocks as well as the header of every other non-intra macroblock and the method of encrypting 64-bit segment from every other macroblock regardless of its type. The last method gives the minimum required processing time with keeping a high level of security.

Master of Science Degree

King Fahd University of Petroleum and Minerals

Dhahran, Saudi Arabia

December 1998

خلاصة الرسالة

الإسم : غسان ابراهيم الرقب

عنوان الدراسة : تقنيات التعمية الجزئية لإرسال آمن للفيديو من نوع MPEG

التخصص : هندسة كهربائية

تاريخ الشهادة : ديسمبر ١٩٩٨م / رمضان ١٤١٩هـ

التعمية الكاملة لمقاطع الفيديو من نوع MPEG تتطلب وقت معالجة كبير جداً. و لإنقاص هذا الوقت مع الحفاظ على مستوى أمني كاف فإن التعمية الجزئية لبعض أجزاء الفيديو هي الطريقة المستخدمة. بعض الباحثين اقترحوا تعمية I-frames فقط. في هذه الرسالة نتائج التجارب ستدلل على أن هذه الطريقة لا تعطي المستوى الأمني المرغوب، ولذلك فإن هذه الرسالة تقترح أربعة طرق للتعمية الجزئية لمقاطع الفيديو من نوع MPEG وذلك لإنقاص وقت المعالجة مع الحفاظ على المستوى الأمني المرغوب. الطريقة الأولى تقترح تعمية ال intra-coded macroblocks في جميع الصور المكونة للفيديو. الطريقة الثانية تدعو لتعمية مقدمات ال P- و B- macroblocks بالإضافة إلى جميع ال intra-coded macroblocks. الطريقة الثالثة تطبق الطريقة الثانية ولكن من خلال التبادل بين ال macroblocks، وبذلك فإن نصف ال macroblocks لا يتم تعميتها. كل من الطريقتين الثانية و الثالثة تعطي مستوى أمني عالٍ مقارنةً بالطريقة الأولى مع زيادة في وقت المعالجة، ولكن وقت المعالجة يبقى صغيراً نسبة إلى التعمية الكاملة. الطريقة الرابعة تعطي مستوى أمني عالي مع وقت معالجة صغير بالنسبة للطرق الثلاث السابقة الذكر. هذه الطريقة تدعو لتعمية ال أربع و ستين bits الأولى من نصف ال macroblocks.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول و المعادن

الظهران ، المملكة العربية السعودية

ديسمبر ١٩٩٨م / رمضان ١٤١٩هـ

Chapter 1

Introduction

1.1 General

Transmission of digital video has become one of the most important aspects of communications for the last decade. Due to huge bandwidth and storage requirements, digital video is compressed before transmission. Many compression algorithms have been developed. The most recent standard that employs these algorithms is the Moving Pictures Expert Group (MPEG) standard. The standard specifies certain syntax for MPEG bit-stream structure. Any disturbance in this bit-stream may corrupt several frames.

Some applications such as military and commercial TV broadcast require secure transmission or storage of the compressed digital video bit-stream. Moreover, video conferencing has become a daily characteristic of financial businesses. It saves time, effort, and travel expenses for large companies. However, the communicated video bit-stream has to be secured against eavesdroppers such as competitors. For this purpose, encryption algorithms can be applied at the transmitter and subsequent decryption algorithms can be applied at the receiver. All cryptographic algorithms require very extensive computations and thus increase the video transmission rate, which has to be, generally, 30 frames/sec. The large processing time that is required for securing an MPEG compressed digital video has created a new research area since 1995.

Previously, researchers dealt with the problem of digital video encryption as a data-encryption problem and they worked on reducing the computation time of the encryption/decryption algorithms. After the launch of the MPEG standard, multimedia and video compression researchers considered the nature of the data being encrypted. The data represents compressed digital video, which has independent and dependent parts. So, if the independent part is encrypted, then the dependent part will be secured inherently. This is called selective encryption. This chapter starts by a discussion of applying scrambling techniques on digital video. Then different aspects of cryptography are discussed in sections 1.3-1.5. Section 1.6 presents a brief discussion about MPEG-I standard.

1.2 Scrambling

1.2.1 The Active Line Rotation principle

Analog video can be disguised through scrambling. The main principle used is called active line rotation. This method has been used since the 1970's [12]. Figure (1.1) illustrates the steps of this process. The part named "Sync." in Figure (1.1) indicates the synchronization point in the video line.

At the transmitter, the video line is cut into two segments at a random cut-point. Thus, the position of the cut-point is varied from one video line to another in the picture. The cut-point is determined by a 60-bit pseudo-random number generator known to both the transmitter and the receiver. After that, the two segments are interchanged and the video line is transmitted. The receiver determines the position of the cut-point and re-interchanges the two segments to restore the original video line [12].

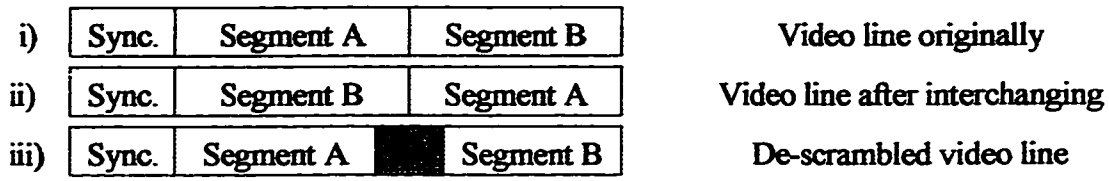


Figure 1.1: The active line rotation principle.

This method has two main disadvantages. The first one relates to the security level. Transmitting the scrambled video line as in part (ii) of Figure (1.1), results in a low security level because of the possible abrupt transition between the two segments A & B. An eavesdropper can notice the abrupt transition and thus determine the position of the cut-

point. The second disadvantage relates to quality. The de-scrambled video line in part (iii) of Figure (1.1) has a shadowed part [12]. This part of the video line is distorted during transmission. This distortion is visible and it degrades the quality of the video considerably. Also, there will be distortion at the edges, but these edges are not visible on normal-TV [12].

1.2.2 The Modified Active Line Rotation principle

To overcome the low security level and low quality of video scrambled by active line rotation, a modified active line rotation was developed in 1983 [12]. Figure (1.2) illustrates the modified method.

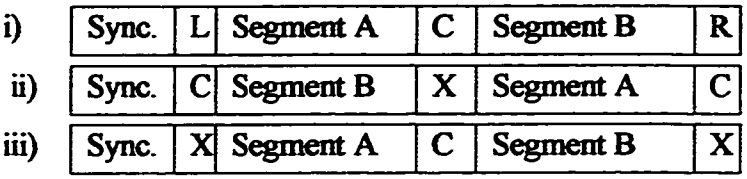


Figure 1.2: The Modified Active Line Rotation Principle.

At the transmitter, the video line is partitioned into two major segments (A & B) and three small segments (L, C & R). Segments A & B are interchanged and segment C is repeated at the two edges as shown in part (ii) of Figure (1.2). Segments R & L are combined in one segment X after multiplying them with Cos^2 *and* Sin^2 curves, respectively. The combination keeps the length of the video line unchanged. In addition, this shaping removes the abrupt transition between the two major segments A & B. Hence, the security level is improved compared to the basic active line rotation principle. Segment C at the start of the video line is shaped by multiplying it with a Sin^2 curve. The C segment at the end of the

video line is shaped by a Cos^2 curve. This shaping protects segment C from distortion. Hence, the quality of the received video improves compared to the received video that is scrambled by the basic active line rotation principle. The de-scrambled line is shown in part (iii) of Figure (1.2). The only distortion is at segment X at both the start and the end of the video line. This distortion is tolerable for TV broadcast because these parts of the video line do not appear on a normal-TV [12].

1.2.3 Applying the Active Line principle to digital video

The active line rotation principle can be modified for application to digital video by interchanging the pixels' values between two segments. This is known as transposition. Transposition of pixels will not degrade the video quality because the values are kept the same and only their positions are changed.

One of two methods can be chosen to transmit the scrambled digital video. The first method is to transmit line-by-line. After receiving each scrambled line, the receiver determines the cut-point position using the pseudo-random generator and then restores the original line. However, the security level of the video transmitted this way is low. The eavesdropper knows that adjacent pixels are highly correlated. So, he will sequentially compare sets of two pixels in the same line until he recognizes the cut-point position. If all pixels in the transmitted line are highly correlated, then the eavesdropper may not be able to determine the cut-point position. However, he does not have to because interchanging the pixels'

positions will not in essence affect the line and it will be essentially the same because its pixels are highly correlated.

The second method of transmitting the scrambled video is by transmitting blocks of the frames in the video. This method is used by multimedia standards. However, since the lines of the video are scrambled, the receiver has to store the entire received blocks until it receives a complete line. Then the line can be de-scrambled. As a result, this method requires a buffer to store the received blocks. This introduces a delay which will adversely affect the transmission rate. Hence, the active line rotation principle is not practical for securing digital video. An alternative method is to use cryptography, which has been used to secure data for decades.

1.3 Brief Review on Cryptology

1.3.1 Cryptology

The science of cryptology has two main branches, *cryptograhpy* and *cryptanalysis*, as shown in Figure (1.3). *Cryptograhpy* deals with developing methods and algorithms to insure security. *Cryptanalysis* deals with methods to break the security systems developed by cryptography. Cryptographers build algorithms to convert a plain text or a video clip into a cipher text or a cipher video clip that is meaningless to the eavesdropper [9].

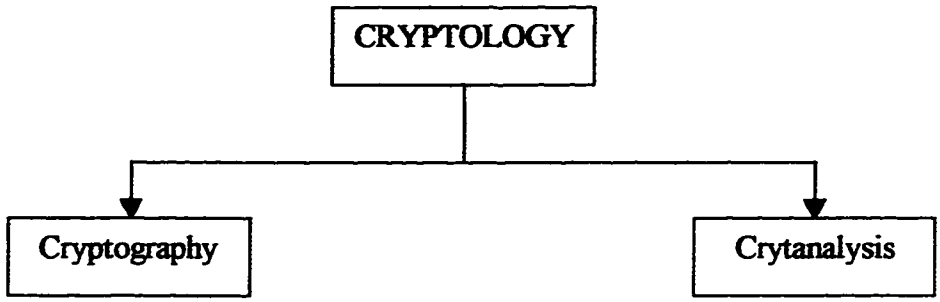


Figure 1.3: Branches of cryptology.

The cryptanalyst uses all the information, all the computational power, all the tricks, and all the technology he has to break the system security. On the other hand, the cryptographer tries to decrease the information available to the cryptanalyst. Also, the cryptographer continuously develops new algorithms and methods, keeping in mind current and future technology, which may be available to the cryptanalyst [26].

1.3.2 Classes of cryptographic systems and applications

There are many encryption techniques. Each technique belongs to a class of cryptographic systems. These classes are communication, file, block, stream, public key and private key encryption. Selection of one of these classes of techniques depends on the application. These classes, together with their applications, will be discussed in this section.

1.3.2.1 Communication Encryption Vs. File Encryption

The decision between communication and file encryption depends on whether the video being encrypted is "in motion" or in storage. "In motion" means that the video is transmitted

over a network consisting of multiple nodes and channel links. If the video is in motion, then communication encryption is appropriate. In the communication encryption class, few seconds are required for secure transmission. This is the time the video clip takes to go from the transmitter to the receiver. On the other hand, in file encryption, video is encrypted for long time storage. As a result, there is enough time for the cryptanalyst to attack the system [9]. File encryption is most appropriate for Video-On-Demand (VOD) systems [4]. There are two sub-classes of communication encryption, namely *link* and *end-to-end* encryption. In link encryption, the video is encrypted and decrypted at each node. This has the advantage of keeping the network secure even if a link is compromised. This technique can be used for military applications, for example, when video clips are sent from the battlefield to military headquarters. These video clips can pass through multiple military stations to update chosen military leaders, until they reach their final destination [9, 10]. The main disadvantage of link encryption is the large time delay created by decrypting and encrypting at each node. Thus, link encryption is not suitable for on-line applications.

In end-to-end encryption, the video clip is encrypted at the sender node and decrypted only at a specific receiver node. The encrypted video passes unchanged through any intermediate nodes. This technique has military applications as well. A video teleconferencing between a general leader and a single sub-leader at the battlefield can be encrypted by this method. The efficiency of this technique is higher than that of link encryption because only two nodes need to be managed.

1.3.2.2 Block Encryption Vs. Stream Encryption

If an algorithm encrypts all the bits in a group simultaneously, then the algorithm is called a *block encryption algorithm*. On the other hand, if an algorithm is applied to each bit separately, it is called a *stream-bit encryption algorithm*.

In block encryption, the video bit-stream is divided into groups containing equal numbers of bits. The key is applied to these groups in a certain process (algorithm) to produce the encrypted video. Block encryption can be practically implemented in software as well as in hardware. Most of the popular encryption algorithms are of the block encryption type. Examples of block encryption algorithms are the Data Encryption standard (DES) and the Rivest-Shamir-Adelman (RSA) algorithm. These two algorithms will be explained in section 1.5.

In stream encryption, a key generator is needed to provide a bit-key to be *XOR*ed with the data bits and produce the cipher stream. This key generator can be a Pseudo-Random Shift Register (PRSR), which is a normal shift register with certain feedback connections. The initialization of the shift register and the feedback connections are considered to be security elements and only authorized individuals know them. The PRSR can be described by a polynomial. The powers of X in the polynomial represent the connections. The X's are just dummy variables [4]. An example of a PRSR is given in Figure (1.4), where $(1+X^2+X^3)$ is the polynomial defined over a Galois Field GF (2).

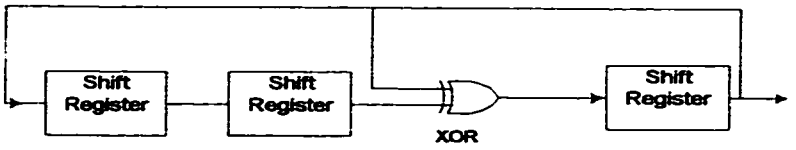


Figure 1.4: A PRSR with polynomial $(1+X^2+X^3)$.

After generating the key, the cipher stream is obtained by *XOR*ing the key with the stream bits:

$$C_i = M_i \oplus K_i; \tag{1.1}$$

where: C_i *is the i - th cipher bit* ,M_i *is the i - th data bit* and K_i is the i - th key bit.

For real-time applications, stream encryption is preferable to block encryption since it is fast. Stream encryption can be easily implemented in hardware. On the other hand, it is difficult and inefficient to implement it in software [10]. Examples of stream encryption algorithms are Rivest Cipher (RC4), Soft-Efficient Algorithm (SEAL), A5 and PIKE [10].

1.3.2.3 Public-Key Encryption Vs. Private-Key Encryption

Witfield Diffe and Martin Hellman have developed the idea of public-key cryptography [10]. In public-key (asymmetric) encryption, two related keys are used. One is used for encryption and is public. The other key is used for decryption and is secret. It is computationally infeasible to compute the secret key from the public key [10]. An example of a public-key encryption algorithm is the RSA algorithm. This algorithm will be discussed in section 1.5 together with the DES, which is a private-key algorithm.

In private-key (symmetric) encryption, the same key is used for both encryption and decryption and it is secret. The main problem with symmetric crypto-systems is key management and security, since the key must be generated by the encryptor and then be distributed to the decryptor. This problem is avoided in the asymmetric crypto-systems because the secret key is known to the decryptor only. This characteristic of public-key cryptography makes it suitable for authentication [9, 10].

Both Figure (1.5) and Figure (1.6) clarify the differences between public and private key encryption algorithms. In general, the asymmetric crypto-systems are slower than the symmetric crypto-systems. For example, the RSA is 1000 times slower than the DES in software implementation and 100 times slower in hardware implementation [10]. In the next section the key management issues are discussed.

1.3.3 Key management

An important aspect of cryptography is key management. In private-key techniques, key management plays a major role in determining the security level of the system because the key is distributed over the network (the channel). Usually, to secure the key when it is distributed, the master key is first generated and is then used to produce the distribution and process keys. The distribution key is used to encrypt/decrypt the process key. The process key is used to encrypt/decrypt the video bit-stream. To maintain key security, the key is generated randomly and changed periodically.

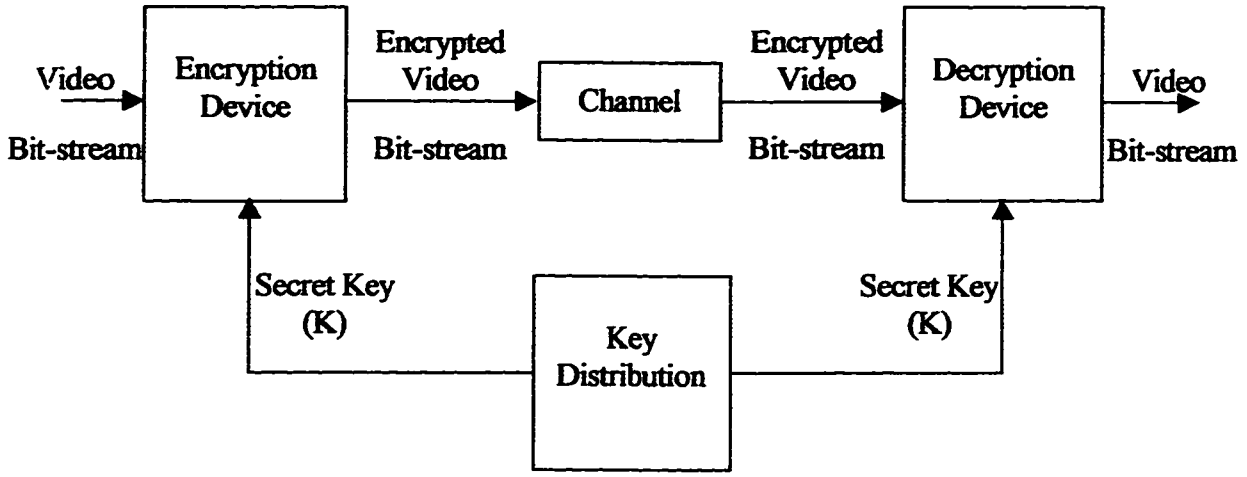


Figure 1.5: Private-Key (Symmetric) Ccrypto-system.

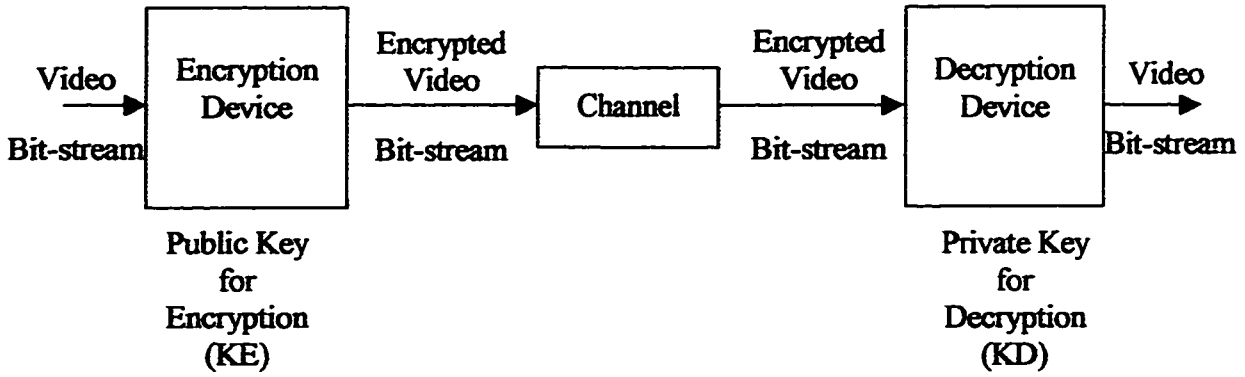


Figure 1.6: Public-Key (Asymmetric) Crypto-system.

Another way to secure the key is to encrypt it and send it to the receiver as user information. Then the receiver applies the decryption algorithm to the encrypted key to get the process key. After that, the process key is applied to the remaining encrypted video bit-stream according to the decryption algorithm.

One way to add more security to the system is to apply different encryption algorithms. One is for encrypting/decrypting the key and the other is for encrypting/decrypting the video bit-stream. For example, the RSA can be used to encrypt/decrypt the key and the DES to encrypt/decrypt the video bit-stream. Although such a method adds complexity to the system, it makes it more difficult for a cryptanalyst to break the security of the system.

Frequently updating the key also improves the security level. However, a mechanism must be developed to keep the receiver synchronized with the transmitter so that the receiver applies the correct key. The receiver can be designed to check the key periodically or the transmitter can set a flag to inform the receiver whenever the key is updated [2].

1.3.4 Brute-Force Attack

If it is assumed that the only way a cryptanalyst can break a secure system is by identifying the key, then all that he needs is a segment of the video bit-stream and the corresponding encrypted bit-stream. By trying all possible keys, he will eventually find the key, which was used in the encryption process. This is known as a "Brute-Force attack" [10]. The cryptanalyst will then be able to decrypt the received video bit-stream until the key is changed. This is one reason why the length of the key is important. As the key length

increases, the average time needed for a cryptanalyst to find the key increases. This is true although the cryptanalyst might find the key from the first trial.

It is possible to calculate the maximum time needed for a "Brute-Force" attacker to get the right key if we know the speed of the machine used. IBM has a new processor with a speed of 900 MHz, and Intel has launched a processor with a bit rate of 10^{12} bits/second. Also, the U.S. National Security Agency (NSA) may have built a very powerful machine specialized in breaking DES systems. (Actually, building such a machine could not have been done without involving one of the large companies specialized in manufacturing processors.) So, in the worst case, a cryptanalyst might have such a machine. If it is assumed that the "Brute-Force" attacker has one of these state-of-the-art machines, then he can test one billion keys per second (1000-Mkeys/second). Thus, the suggested machine can test all of the possibilities of a 40-bit key (i.e. 2^{40} keys) in: $\frac{2^{40}}{10^9} = 18.3$ *minutes*. If the key is a DES key and has a length of 56 bits, then the time needed to identify the key is: $\frac{2^{56}}{10^9} \times \frac{1}{3600} = 834 days$.

If 2000 machines work in parallel and have the same operating capabilities as above, then the time needed to identify the 56-bit key is 10 *hours*.

For convenience, let the cost of each machine be US$1000. So, the cost of using 2000 machines to identify the 56-bit key in ten hours is two million dollars. Table (1.1) summarizes the processing time and the corresponding cost needed for different key lengths. The general formula used is:

$$Time = \frac{2^k}{L \times n \times 3600} hours \tag{1.2}$$

$$Cost = n \times c \tag{1.3}$$

where:

n : # of machines used,
L : # of keys tested per second by one machine,
k : key length in bits, and
c : cost of each machine.

To keep these calculations general for any machine that can test L keys in one second, L will be kept as a variable in equation (1.2). Also c, which is the cost of a single machine, will be the variable in equation (1.3). Table (1.1) can be generalized as summarized in Table (1.2). As expected, for a certain key length, as the machine capability increases, the average time to break the system decreases and the cost increases.

		Key Length in Bits			
No. of used machines	**Cost (US$)**	**40**	**56**	**64**	**128**
1	1000	18.3 min	834 days	585 years	12×10^{24} years
1000	1 M	1.1 sec	20 hours	213 days	12×10^{18} years
10,000	10 M	0.12 sec	2 hour	21.3 days	12×10^{17} years
100,000	100 M	10 m sec	12 min	2 day	12×10^{16} years
1,000,000	1 G	10 μ sec	72 sec	5.2 hours	12×10^{15} years

Table 1.1: Time and cost needed to identify the key by Brute-Force attack for a machine that can test 1-G keys/sec.

		Key Length in Bits			
No. of used machines	**Cost (U$)**	**40**	**56**	**64**	**128**
1	C	35350/*L* years	2.3/*L* G years	585/*L* G years	$1.2\times10^{31}/L$ years
1000	1000×c	35.35/*L* years	2.3/*L* M years	585/*L* M years	$1.2\times10^{28}/L$ years
10,000	10,000×c	3.5/*L* years	230/*L* K years	585/*L* M years	$1.2\times10^{27}/L$ years
100,000	100,000×c	4.2/*L* months	23/*L* K years	5.85/*L* M years	$1.2\times10^{26}/L$ years
1,000,000	1,000,000×c	12.73/*L* days	2.3/*L* K years	585/*L* K years	$1.2\times10^{25}/L$ years

Table 1.2: Time and cost needed to identify the key by Brute-Force attack for a machine that can test L keys/sec; c is the cost of one machine.

The question that arises here is how important are these video clips so that a cryptanalyst would pay all this money to get them? The answer depends on the application. If the application is for very critical military purposes, for example, then it may be justified to pay a lot of money to get these video clips. Another important thing to note is that this analysis is based on the current available computing capability and what takes hours or days by today's technology might take minutes by future technology.

1.4 Mathematical Background

In this part, the mathematical basis for a secure system is studied according to Shannon's theory of secrecy systems. Suppose that the compressed video bit-stream consists of n finite

blocks $M_1, M_2, \ldots M_n$. The apriori probabilities of these blocks are $p(M_1), p(M_2) \ldots p(M_n)$, respectively. When the encryption algorithm is applied to these blocks, the output will be another group of blocks, $C_1, C_2 \ldots C_m$ given by:

$$C_m = T_i \{M_n\} \tag{1.4}$$

where T_i is the encryption algorithm operation. The cryptanalyst will receive a single C_i and he will try to calculate the aposteriori probabilities of the various blocks $P(M_n/C_i)$ and interpret the received block that gives high aposteriori probability. If the aposteriori probabilities are equal to the apriori probabilities, the information available to a cryptanalyst is zero and the system is perfectly secure. This case does not exist in reality and always the cryptanalyst has some information about the transmitted blocks and the keys used in the encryption algorithm. At least he can guess them. For these reasons this definition of secrecy systems is impractical [1,11].

It has been found by Shannon [11] that the condition for secrecy systems depends on the unicity distance. For a cryptanalyst to break a cryptography system, all that is needed is the encrypted video bit-stream and the key. Now, some terms will be defined to help in finding out the condition of a practical secure system:

- Let the equivocation of the key be $H(K/C)$. It represents the provided information about the used key K given that the encrypted video bit-stream C is known. As $H(K/C)$ approaches zero, the system becomes less secure.

- Define the smallest amount of video bit-stream needed to make the key equivocation zero to be the unicity distance U. It is a statistical measure of how much an encrypted video bit-stream has to be available for a cryptanalyst to have a reasonable solution for the key.

- Let the video bit-stream M have a length of N bits.

- Suppose that each pixel in the uncompressed video is represented with L bits. Then the maximum amount of information contained in each pixel of the original video is $r_a = L$. r_a is known as the absolute rate of the original video. However, the true rate r_t is defined as $r_t = \frac{H(M)}{n}$ where n is the number of blocks in the video bit-stream. Hence, the redundancy D in the video is:

$$D = r_a - r_t \tag{1.5}$$

Shannon has shown that the unicity distance is given by:

$$U = \frac{H(K)}{D} \tag{1.6}$$

If the key is chosen completely at random to maximize the unicity distance, then $H(K)$ equals the length of the key L_k. Hence,

$$U = \frac{L_k}{D} \tag{1.7}$$

The above result indicates that decreasing the redundancy in the compressed video increases the unicity distance. This in turn increases the security level of the system. A perfectly secure transmission can be achieved using perfect compression that reduces the redundancy to zero. Unfortunately, such compression techniques are not available, and it is unlikely that they will ever exist due to the high correlation between adjacent pixels and also adjacent frames. The above result also indicates that the more compression is applied to the original video the less encryption is needed on the compressed bit-stream in order to achieve a desired level of security [1,2,10,11,18]. The general system that achieves this compromise is depicted in Figure (1.7).

According to Shannon's definition of secrecy systems, a secure video system is that in which the encrypted video bit-stream carries no information other than the length of the original video bit-stream. This can be satisfied when the key is not reused in the same video bit-stream. This means that if a key is applied on a block, it should not be reapplied on another block in the same video bit-stream [11]. This is impractical.

Usually, video clips have a lot of redundancy and each frame is obtained from the previous one by, for example, translation in the plane, rotation, or deformation. This affects the encryption algorithm from two sides. First, when encrypting the whole stream with the same key, many blocks in the video bit-stream will be repeated and this gives the cryptanalyst an idea about the video. Second, this redundancy degrades the security level of the system as found by Shannon. Applying a good compression algorithm to the video before encrypting it will reduce the redundancy and that will increase the security level. Thus, compression

provides two main benefits to secure video systems. It reduces the size of the video to be encrypted and hence reduces the processing time. In addition, it improves the security level as mentioned previously [2, 11, 18].

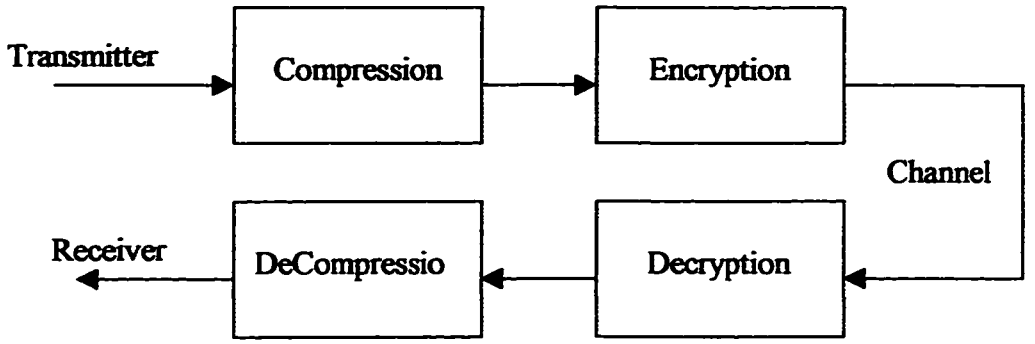


Figure 1.7: A typical secure video system.

1.5 Popular Cryptographic Algorithms

The most popular crypto-algorithms used in practical systems are the DES and the RSA [10]. An explanation of each algorithm is introduced next in addition to a comparison between the two.

1.5.1 The Data Encryption Standard (DES)

The Data Encryption Standard (DES) was invented by IBM in 1977. The U.S. National Security Agency (NSA) used it and imposed restrictions on both implementing it and explaining the methods DES applies. The general method is well explained in the literature

and it is introduced here in brief. The DES is a private-key (symmetric) crypto-system and is applied on blocks of the video bit-stream. It applies both the substitution and the transposition methods. DES is available as both a hardware chip and a software program [10].

DES uses a 64-bit key, 56 bits of which are actually used and the remaining 8 bits of which are used for parity check. The input block of the DES algorithm is 64 bits long. First, the key is permutated and 56 bits are chosen as the master key. Then, these 56 bits are shuffled into 16 different configurations, each 48 bits long. This shuffling process applies shift in a certain way. These sixteen different configurations are called sub-master keys. Figure (1.9) clarifies this process. The C_i's and the D_i's in the figure represent left circular shifts of the input 28-bit blocks. The first permutated choice 2 block gives the first sub-master key K1, which is used in the first stage of the DES, and so on up to the last stage, which is stage number sixteen as shown in Figure (1.10).

The 64 bits that make up the input data block are permutated by a table called the Initial Permutation (IP) table. Then they are divided into two parts, each 32 bits long. One of these two parts, which is the right-hand one, is passed as is to the next stage as the left-hand part of the new stage. In addition, it is replaced by a 48-bit word from a special table, called the E-table. This 48-bit word is *XOR*ed with the chosen 48-bit sub-master key. The result is a 48-bit word, which is converted to another 32-bit word by a special box called the S-box. The output of the S-box is permutated by a P-table to produce a 32-bit word. Now, this 32-bit word is *XOR*ed with the untouched left part of the input block (32-bit word also). The result

is the right-hand part of the next stage. This completes the explanation of the main block of the DES algorithm. It is called the Standard Building Block (SBB). The function of the building block is shown in Figure (1.8).

This SBB is repeated sixteen times and at each time a different sub-master key is used. After the last stage (stage number 16), the two 32-bit words are recombined by a table called the Final Permutation (FP) table. The output is a 64-bit block. The whole block representing the DES is shown in Figure (1.10). The block, in the figure, that is indicated as $f(R_i, K_{i+1})$, (i = 0 .... 15), is a function that takes the right part of the 64-bit input word and the key for that stage (from Figure (1.9)) and applies on it the function of the SBB discussed above. The same algorithm is used for both encryption and decryption. The IP-, S- and the P-tables are matrices that do permutation and substitution for the input blocks [1, 10].

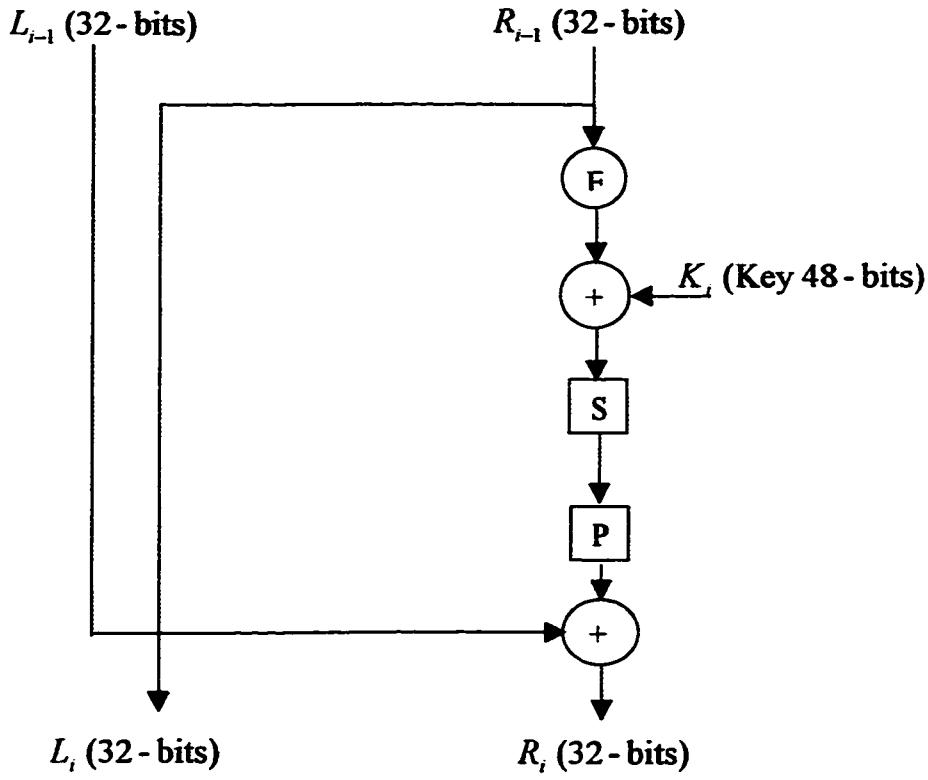


Figure 1.8: The Standard Building Block (SBB).

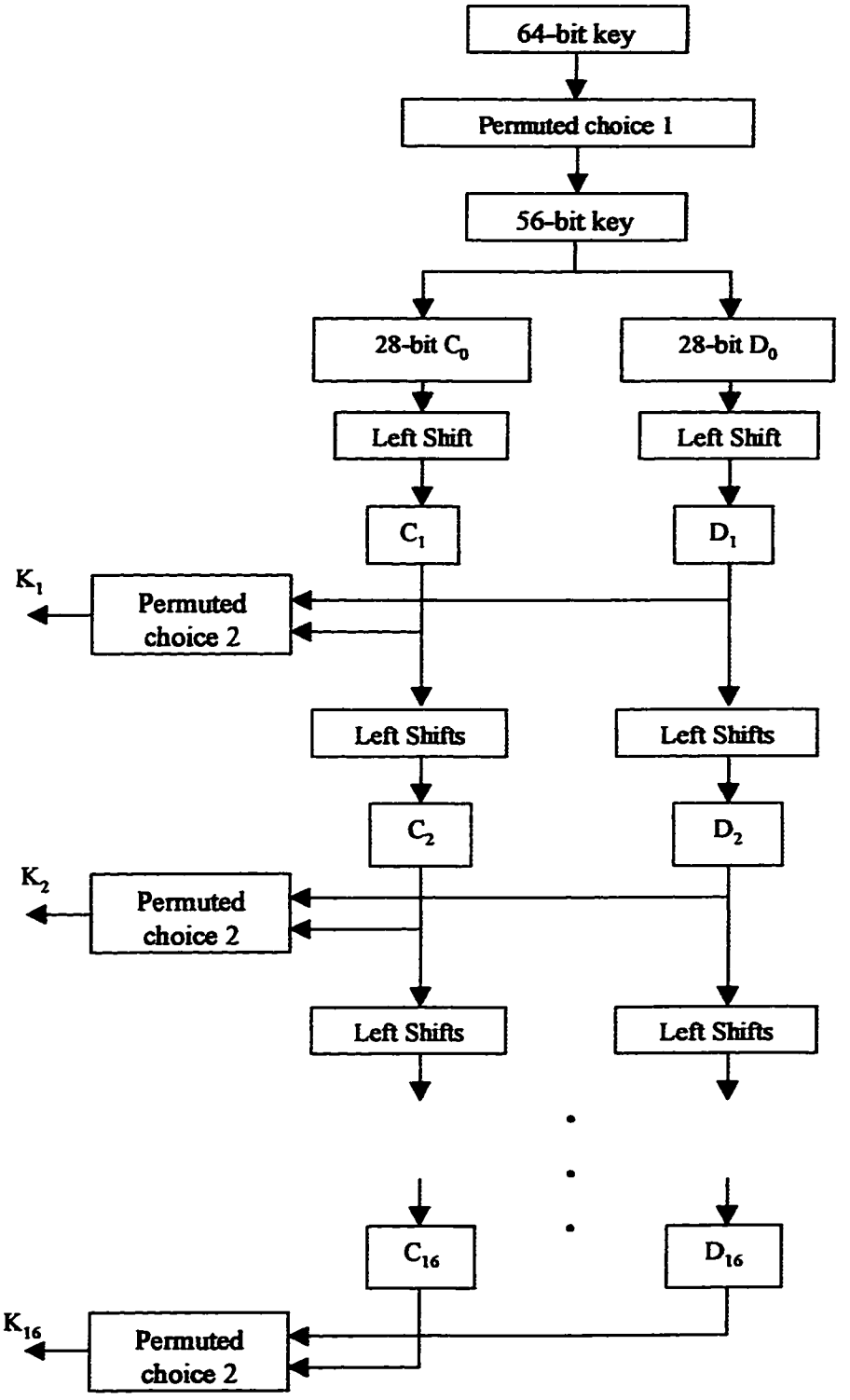


Figure 1.9: The DES-Key shuffling process

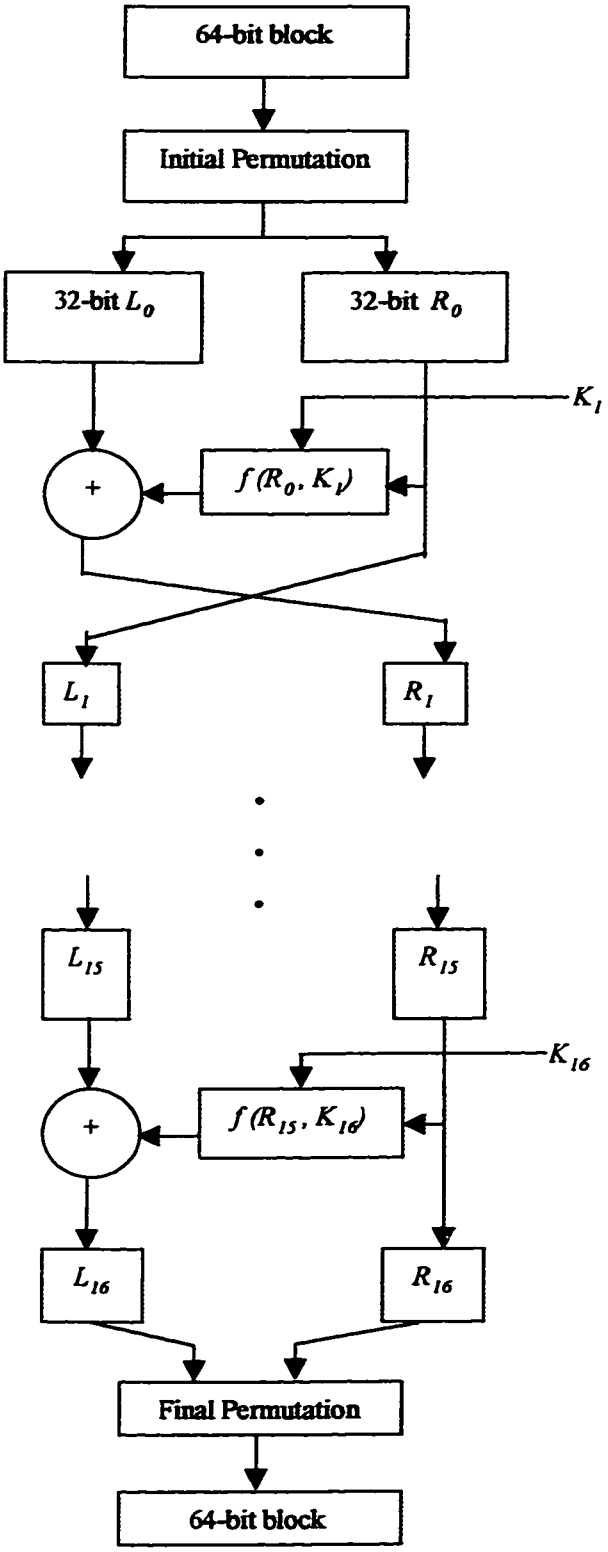


Figure 1.10: The DES algorithm.

To gain some insight on the processing time of encrypting a digital video clip, assume that there exists a machine which can encrypt L_{DES} DES blocks in one second. Also, assume the frame size of the video clip to be $n \times m$ pixels and that each pixel is represented by 24 bits. Hence:

$$\text{Number of DES blocks in one frame} = n \times m \times \frac{24}{64} \text{ DES blocks,} \tag{1.8}$$

$$\text{Processing time for one frame} = \frac{n \times m}{L_{DES}} \times \frac{24}{64} \text{ seconds,} \tag{1.9}$$

$$\text{Processing time for 30 frames} = \frac{n \times m}{L_{DES}} \times \frac{24 \times 30}{64} \text{ seconds.} \tag{1.10}$$

Usually the frame size of an MPEG video clip is either 352×240 or 720×480 pixels. It is given by Schneier [10] that a (486) PC can encrypt 43,000 DES blocks in one second. This means that a 266 MHz-Pentium PC can encrypt $(43{,}000 \times 4 \times 2)$ DES blocks in one second. The 4 factor is due to the speed, while the 2 factor is due to the Pentium technology. So, if we have a video that consists of frames of size 352×240, the Pentium-PC can encrypt one frame in 0.092 second, assuming that each colored-pixel is represented by 24 bits. So, the processing time for the 30 frames is 2.76 seconds. If the frame size is 720×480, then the processing time is 11.3 seconds. Hence, encrypting the whole video bit-stream is inefficient and it degrades the system considerably. These processing time calculations are summarized in Table (1.3). The transmission rate has to be generally 30 frames/sec. It can be seen from

Table (1.3) that the encryption process alonerequires much more than one second to encrypt one second of video (30 frames) in addition to the encoding time.

Machine	Frame Size	Processing time of one frame (seconds)	Processing time of 30 frames (seconds)
486-PC	352×240	0.737	22.10
486-PC	720×480	3.014	90.419
266-Pentium-PC	352×240	0.092	2.76
266-Pentium-PC	720×480	0.377	11.30

Table 1.3: Processing time for encrypting video frames using DES.

1.5.2 Rivest-Shamir-Adelman (RSA) algorithm

Rivest, Shamir, and Adelman designed a block public-key crypto-system in 1978. The new algorithm derives its name from the designers' initials (i.e. RSA). It implements the theory of the trapdoor one-way function. This is a function that you can go in one direction, and it is computationally infeasible to go in the other direction. For example, if the function is $f(x) = x^7 + x^5 + 2x^3 + 6x^2$, then given x, $f(x)$ is easily computed. On the other hand, given $f(x)$, it is difficult to get x.

RSA uses the idea that factorizing the product of two large prime numbers is very difficult. The larger these two primes are, the more secure the system is. On the other hand, the larger the primes are, the slower the algorithm is. This introduces limitations on the usage of RSA in real-time applications. The following steps describe how the algorithm works.

- Given M, the value of the block to be encrypted.

- Choose two large prime numbers p and q.
- Get $n = p \times q$.
- Get $\Phi(n) = (p-1) \times (q-1)$.
- Choose a prime number d that satisfies two conditions:

  - d is larger than both p and q.

  - d must have no common divisor with $\Phi(n)$.

- Compute e such that $e \times d = 1 \pmod{\Phi(n)}$.
- The encrypted block is computed by: $C = M^e \pmod{n}$.
- To get the block value back: $M = C^d \pmod{n}$.

Thus e is the encryption key and it is public while d is the decryption key and it is private. Also, p and q are secret. So, once e and d are obtained from p and q, these latter two quantities must be discarded. In addition, n is known but since it is the product of two large (at least 200 decimal digits each) prime numbers, it is difficult to factorize. The standard key used in the RSA algorithm is 128 bits long and can be extended to 512, 768, 1024, 2048 bits [1, 10]. The next example clarifies the algorithm and verifies its applicability.

If the codes of three blocks to be encrypted are: $M_1 = 687, M_2 = 232, M_3 = 668$, then:

- Choose p=47, q=71.
- $n = p \times q$ =3337.

- $\Phi(n) = (p-1)\times(q-1) = 3220.$
- Choose d such that it satisfies the conditions given above; let d=1019.
- $e = d^{-1} \mod \Phi(n) = 79.$
- $M_1 = 687 \Rightarrow C_1 = (687)^{79} \mod 3337 = 2091.$

  $M_2 = 232 \Rightarrow C_2 = (232)^{79} \mod 3337 = 2756.$

  $M_3 = 668 \Rightarrow C_3 = (668)^{79} \mod 3337 = 2423.$
- To get the block values back:

  $C_1 = 2091 \Rightarrow M_1 = (2091)^{1019} \mod 3337 = 687.$

  $C_2 = 2756 \Rightarrow M_2 = (2756)^{1019} \mod 3337 = 232.$

  $C_3 = 2423 \Rightarrow M_3 = (2423)^{1019} \mod 3337 = 668.$

This returns the original blocks. The part that takes a large time is computing the encryption key *e*. Many researchers had suggested different methods to speed up the RSA algorithm. The method used to find the *Ci*'s and *Mi*'s in the above example is used by Sklar [1].

RSA is widely used for authentication. The transmitter decrypts the video by the secret key *d*. Then the receiver encrypts the received video by the public key *e*, which gives the original video.

Doing similar calculations as the ones done in Table (1.3) for DES, the resulting calculations for the processing time of RSA are listed in Table (1.4). As mentioned by Shneier [10], the RSA is 1000 times slower than DES in software implementation. So, for applications such as commercial TV broadcast and the military, DES is preferable to RSA in terms of speed. The RSA is preferable to the DES regarding key management and distribution. So, a way to utilize the benefits of both algorithms is to encrypt the DES-key by RSA and the video bit-stream by DES. Encrypting the key by RSA will not affect the speed of transmission if the period of changing the DES key is long. On the other hand, if the DES key is changed quite often, using RSA for encrypting the process key adversely affects the processing time.

Machine	**Frame Size**	**Processing time of one frame**	**Processing time of 30 frames**
486-PC	352×240	12.283 min.	6.142 hours
486-PC	720×480	50.233 min.	25.117 hours
266-Pentium-PC	352×240	92 sec.	46 min.
266-Pentium-PC	720×480	6.283 min.	3.142 hours

Table 1.4: Processing time for encrypting video frames using RSA.

Over 20 years, DES has shown its ability to ensure security [10]. It satisfies the security level required by very sensitive applications. Such applications include the military and commercial-TV broadcast. Table (1.5) indicates that a 64-bit DES key ensures a reasonable security level. This is judged to be reasonable for very sensitive applications.

	DES	RSA
Inventor	IBM in 1977	Rivest-Shamir-Adelman
Type	Private-key (Symmetric)	Public-key (Asymmetric)
Block Size	64 bits	128 to 512 bits
Key-Length	64 bits (8 of them for parity check)	128, 512, 768, 1024 .... bits
Security lies in	The key	How large the chosen prime numbers are
key management	Important	Simple
Speed	Fast	Slower than the DES by 1000 times in software implementation
Flexibility	Flexible and suitable for improvement; double and triple DES can be implemented on the old DES [25]	Fixed procedure
Most Popular Application	Encryption and Decryption	Authentication
Limitations	NSA puts some restrictions on improving and using the system	Any application of RSA must have permission from RSA Data Security Inc.

Table 1.5: Comparison between the DES and the RSA algorithms.

1.6 MPEG Video Compression Standard

Although many proprietary compression algorithms have been developed in the past two decades, MPEG-I and MPEG-II are the standard algorithms. Both standards were developed by a special study group known as the Moving Pictures Expert Group (MPEG). This group was established in 1988 and jointly sponsored by the International Standards Organization (ISO) and the International Electro-technical Commission (IEC) Joint Technical Committee (JTC 1) on information technology. The group came up with the first member of the MPEG family, MPEG-I, in November 1992, and introduced MPEG-II three years later. Both of these standards apply lossy compression techniques in order to achieve high compression ratios that are suitable for the intended applications; MPEG-I is the video compression standard for Compact Disk (CD) -based multimedia applications. The structure of MPEG-I bit-stream can be found [17].

The MPEG standard leaves many degrees of freedom for manufacturers to design their own MPEG encoders. It does not specify the decoder design but, rather, specifies the bit-stream syntax. Chariligions, who is considered the father of MPEG, has been quoted saying "It must be borne in mind that MPEG standards do not specify complete systems" [20]. Actually, this is what makes MPEG a generic standard and one that is not expected to quickly become obsolete [7, 8, 13-16, 19-22].

The second member of the MPEG family is MPEG-II standard. The main application of MPEG-2 is TV broadcast. Some of the conspicuous applications of MPEG-II are TV-

Broadcast, direct broadcast satellite (DBS), digital versatile disk (DVD) and high-definition TV (HDTV) [43]. Both standards MPEG-I and MPEG-II have a similar core, and their bit-stream structures are also similar. Similarly MPEG-IV has the same main structure as MPEG-I but with more functionality. The main difference in the development of these standards is that MPEG-IV is developed for several applications not one as the other two standards. Hence, it can support wide variety of bit rates [42-44].

MPEG-IV main application is for Mobile usage [44]. The main functionalities MPEG-IV has are [43]:

- Efficiency in coding different media data such as text, video, audio and graphics.
- Error resilience for error-free transmission over noisy channels as the wirless ones.
- The ability to interact with the audiovisual scene generated at the receiver.
- The ability to encode shaped video objects.

These features make MPEG-IV very suitable for Mobile applications. Also, in 1997, a group has been initiated to develop MPEG-7. In this thesis, the selective encryption techniques will be limited for MPEG-I. Accordingly, they can be applied to all other MPEG standards since they all have the same main structure as MPEG-I.

1.6.1 MPEG-I Bit-stream Structure

A typical MPEG-I bit-stream consists of several layers as depicted in Figure (1.11). The function of these layers is mainly for synchronization of audio and video sources. The first layer is the system layer at which the bit-stream is structured into packs of various lengths.

The audio, video, and system information are multiplexed within each pack [17], and each pack is structured into a number of packets. Some packets contain system information, some packets contain audio information, and other packets contain video information. Since this thesis deals with the MPEG-I video bit-stream, the following discussion will be limited to the video bit-stream and will exclude the audio bit-stream.

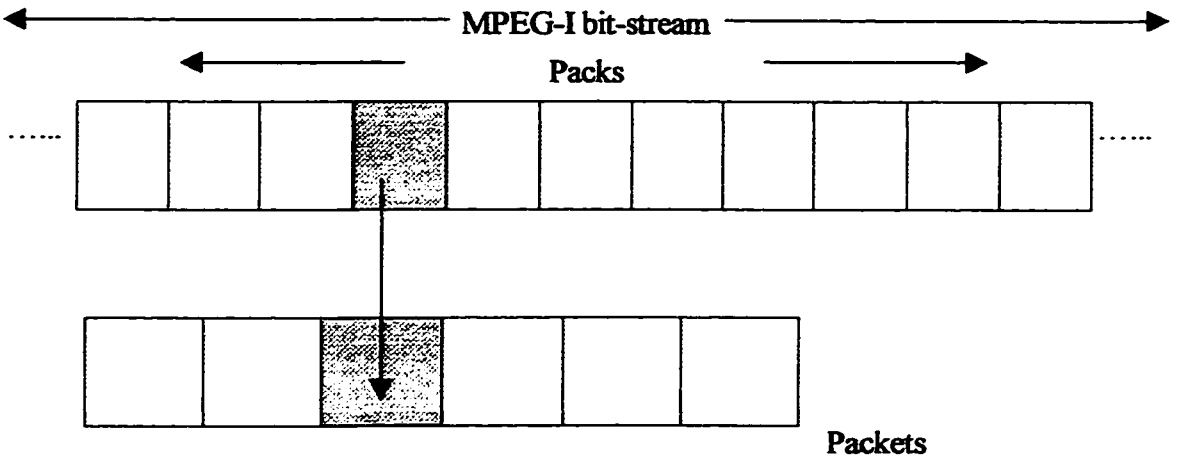


Figure 1.11: The main layers of an MPEG-I bit-stream.

1.6.2 MPEG-I Video Bit-stream Structure

The structure of an MPEG-I compressed video bit-stream consists of consecutive layers. These layers are the video sequence, the group of pictures (GOP), the picture, the slice, the macroblock (MB) and the block layer. Table (1.6) lists the start code and function of each layer [13, 17]. All headers in the bit-stream, except the macroblock (MB) header, are byte aligned. Figure (1.12) depicts the MPEG-I video bit-stream structure in two different ways.

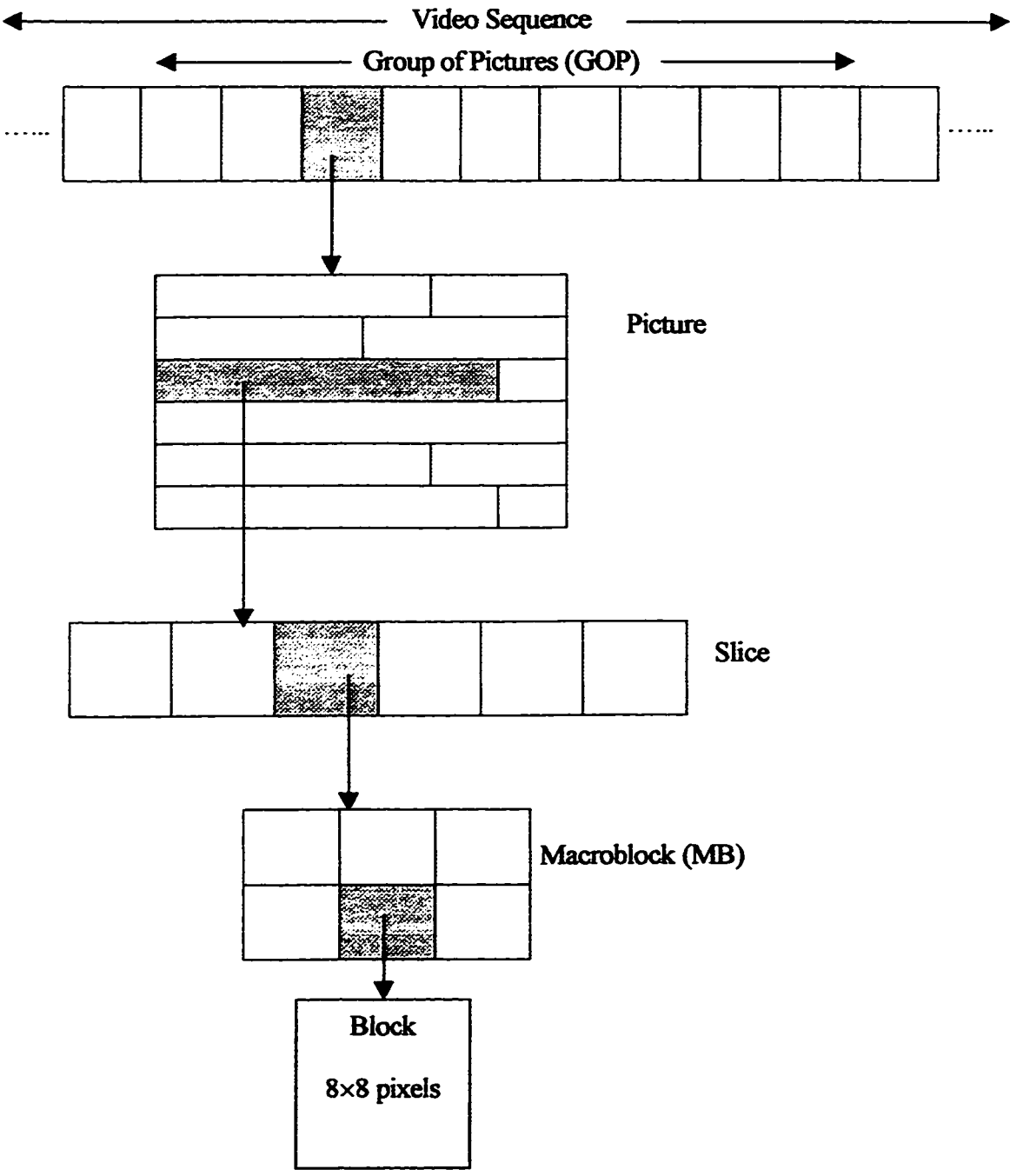
There are several advantages to having a layered structure for the MPEG-I bit-stream. An MPEG-I bit-stream has many entities that are logically distinct. Using layered structure separates these entities. Furthermore, layered structure prevents ambiguity and facilitates the decoding process. In addition, it makes MPEG-I generic, flexible and an efficient algorithm [8].

Layer	Start Code	Function
Sequence Header	000001B3 Hex	Has one or more GOP's
GOP	000001B8 Hex	Random access in the sequence
Picture	00000100 Hex	Primary coding unit
Slice	00000101 through 000001AF Hex	Resynchronization unit
Macroblock (MB)	Does not have	Motion compensation unit
Block	Does not have	Discrete Cosine Transform (DCT) unit

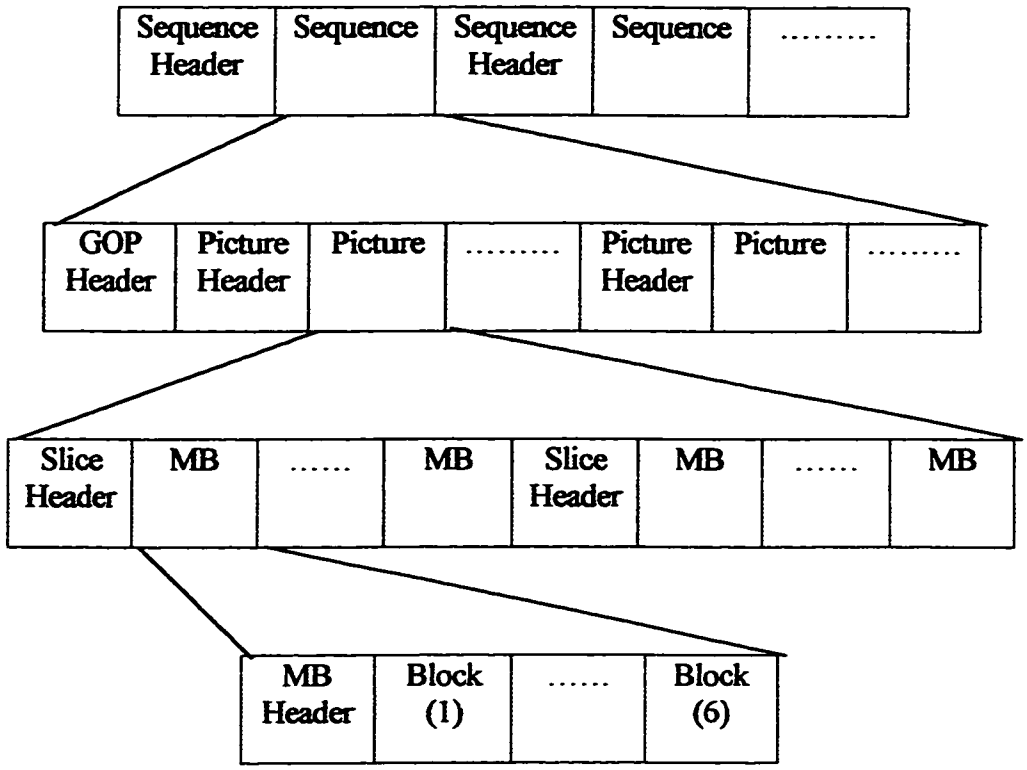
Table 1.6: Functions and start codes of the layers in an MPEG-I compressed video sequence.

1.6.2.1 Video Sequence

The first layer in the MPEG-I compressed video bit-stream is the video sequence. Each video sequence starts with a sequence header, which contains a sequence start code, horizontal and vertical sizes of each frame, pel aspect ratio, bit rate, Video Buffering Verifier (VBV), the intra- and the non-intra quantizer matrices, and some user data [13, 17]. The VBV is used to determine the minimum required buffer size to prevent overflow and underflow at the decoder. The user data can be used by the user to transmit non-standard parameters. Also, each video sequence ends with a special code known as the ISO-end code.


Video Sequence
Group of Pictures (GOP)
Picture
Slice
Macroblock (MB)
Block
8×8 pixels

(a)



(b)

Figure 1.12: Layers of MPEG video bit-stream by looking at them through: (a) the displayed video clip and (b) the bit-stream. (GOP indicates a group of pictures; MB indicates a macroblock.)

1.6.2.2 Group of Pictures

Each video sequence is composed of several groups of pictures (GOP). Each of these groups of pictures starts with a header that is followed by the data of the GOP. The header of the GOP contains the GOP start code, a time code, the closed GOP bit and some user data [13, 17]. The closed-GOP bit indicates whether the current GOP is closed or open. A GOP is closed when there is no prediction required from another GOP. It is open when prediction is required from outside the GOP. As in the sequence header, the user data can be used to transmit non-standard parameters.

1.6.2.3 Picture

Each GOP is composed of several pictures, and each picture starts with a picture header followed by the picture data. The picture header contains the picture start code, the picture temporal reference, picture coding type, VBV (Video Buffering Verifier) delay, and some information to indicate the resolution of the encoded motion vectors. The picture temporal reference indicates the display order of the picture. The VBV gives the time required to fill the buffer of the current picture before starting decoding. It is used with the bit rate to control the display time of each picture.

There are three kinds of frames in an MPEG-I compressed video bit-stream. They are intra-coded I-frames, motion-compensated forward predicted P-frames, and motion-compensated bi-directional predicted B-frames. The I-frames are encoded without referring to any other frame. On the other hand, P- and B-frames are encoded with respect to another reference

frame. Motion compensation (MC) techniques are used to predict the P-frame from the last I- or P-frame. Similarly, the B-frames are predicted either from the last I- or P-frame or the next P- or I-frame. This is shown in Figure (1.13). For P- and B-frames, only the prediction error and the motion vectors are encoded and transmitted. Due to the high correlation between the frames in the video, encoding a P- or B-frame requires much fewer bits than encoding an I-frame [8, 13, 17].

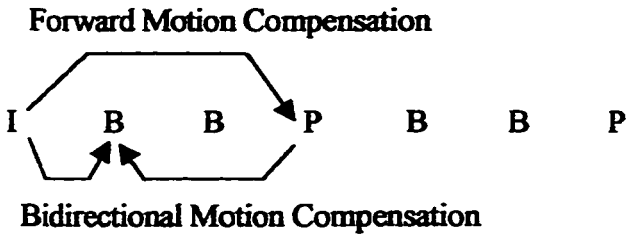


Figure 1.13: A pictorial representation of predicting the P- and B- frames.

In a typical GOP there will be a number of consecutive B-frames. A typical B picture is obtained (or, as it is referred to in the standard, bidirectionally predicted) by interpolating from the reference pictures. As a result, increasing the number of consecutive B-pictures increases the interpolation error, and the correlation between the reference pictures decreases. This reduces the quality of the video. The MPEG committee suggested two or three consecutive B-pictures and a GOP of 10 or 15 pictures. This was determined by running some tests and evaluating the resultant video clips [8, 13, 17].

1.6.2.4 Slice

Each picture or frame is comprised of one or more slices. The number of slices in a picture range from one slice to a number equal to the number of macroblocks in the picture. It is left for the user to specify the number of slices in a picture. Since the slices are used for synchronization purposes, a large number of slices should be used with noisy channels. The slice header contains the slice start code and the quantizer scale [8, 13, 17].

1.6.2.5 Macroblock

Each slice contains any number of macroblocks (MB's). The MB data is preceded by a header that contains information about its type and the Motion Vectors (MV) that are used in Motion Compensation (MC) techniques. Some of these parameters are:

- Macroblock stuffing, which is used when there is underflow in the buffer.
- Macroblock address increment, which has a value between 1 and 33 and is used in assigning the codewords.
- Macroblock type, which can be I, P or B. Motion Compensation (MC) techniques are used to predict P and B MB's. In some P- or B-frames, MC fails to produce good prediction of some macroblocks. These macroblocks are encoded as independent of the reference frames. Therefore, I-frames contain I-macroblocks, P-frames contain both I- and P-macroblocks, and B-frames contain I-, P- and B-macroblocks [7, 8, 13, 14, 15, 17].
- Quantizer scale.

- Motion horizontal/vertical forward/backward code.

- Coded Block Pattern (CBP).

Each macroblock contains six blocks. Each block is 8×8 pixels and contains the Discrete Cosine Transform (DCT) data. Four of these 6 blocks are luminance blocks and the other two blocks are chrominance blocks. Each block has an end of block code [7, 13, 17].

The output of the MPEG-I algorithm is well constructed and any disturbance or error in the stream will disturb the whole video. This makes encryption of MPEG-I compressed video a very difficult task.

1.7 Statement of the Problem

Since MPEG-I compressed video bit-streams are typically huge and data encryption/decryption algorithms are relatively slow, using traditional encryption techniques in order to secure the transmission of MPEG-I compressed bit-streams tremendously increases the over all processing time. Several researchers have attempted to reduce this processing time through the encryption of selected portions of the MPEG-I video bit-stream. They assume that with proper selection of these portions, the entire bit-stream can be secured at a reasonable processing time.

In this thesis, the security level of several selective encryption techniques for use with MPEG-I bit-streams will be evaluated empirically. Two new selective encryption techniques

will be developed to achieve a higher security level. The first technique calls for encrypting a certain number of I-macroblocks in all frames (regardless of the frame type). The second technique calls for encrypting the P- and B- macroblocks' headers in addition to encrypting a certain number of the I-macroblocks in all frames (regardless of the frame type). A secure method for key management will be devised for the developed selective encryption/decryption techniques, and the achieved security levels of both techniques will be analyzed. The processing time achieved by both techniques will be compared to the full encryption case.

Chapter 2

Encrypting the I-Frames of MPEG-Compressed Video Bit-streams-An Evaluation

2.1 Introduction

There are many applications requiring secure transmission of video clips. The required security depends on the sensitivity of the information in these applications, varying from extremely high to relatively low. Some of these applications also require real-time transmission. This thesis addresses sensitive applications that require real-time transmission of digital video clips. Military and commercial TV broadcast are examples. A suitable, secure transmission system for these applications is depicted in Figure (1.7). In such a system, the

digital video clip undergoes compression and encryption at the transmitter and decryption and decompression at the receiver. Real-time transmission requires that to be accomplished in less than 1/30 of a second per frame. Usually, the MPEG-I compression standard is used for the compression/decompression stages. Since MPEG-I-compressed video bit-streams are typically huge and current data encryption/decryption algorithms are relatively slow, using these encryption techniques tremendously increases the over all processing time, exceeding the standard 1/30 of a second per frame.

Some researchers [5, 6, 18, 28] have proposed selective encryption/decryption in order to reduce the overall processing time. In selective encryption only the I-frames in the bit-stream are encrypted. It is assumed that the remaining frames inherit their security from the encryption of the I-frames. This chapter analyzes and evaluates the security level achieved by selective encryption. Section 2.2 presents a summary of the current research on the encryption of compressed digital video. Section 2.3 presents the experimental set up, the methodology, and the characteristics of the test video clips used in this thesis. In Section 2.4, the processing time needed for encrypting only I-frames is compared to the processing time needed for encrypting the entire MPEG-I video bit-stream. Section 2.5 analyzes the suggestions found in the literature for improving the security level of the proposed I-frame encryption method.

2.2 Research trends in securing digital compressed video

Several researchers have tackled the problem of encrypting compressed images or stream of images (video). Al-Jabri and Al-Asmari [3] apply the DES algorithm to JPEG (Joint Pictures Expert Group) compressed images. They divide the bit-stream into three blocks (*A*, *B* and *C)* of decreasing importance, in terms of security. Each block is a multiple of *m* bits. Two sub-keys k_b and k_c are generated from the master key k_a. The researchers presented two methods of generating the sub-keys. One is based on block ciphers while the other is based on number theoretic algorithms. The latter method is known as the discrete logarithm problem, where a large prime number ρ is chosen together with its generator element α. Then the two sub-keys are generated according to the following equations: $k_b = \alpha^{k_a} \mod \rho$ and $k_c = \alpha^{k_b} \mod \rho$.

Block *A* is divided into sub blocks. Each has *m* bits. These sub blocks are encrypted by the three keys k_a, k_b and k_c. The same procedure is applied to block *B*, but with keys k_b and k_c only. Finally, the key k_c encrypts block *C*. In this technique, full encryption is applied to the entire bit-stream. However, an emphasis is placed on the important part, which is block *A*. This gives a high level of security, but with low efficiency, because blocks *A* and *B* are encrypted more than once. In addition, there is a delay created by generating the sub keys and applying them to blocks *B* and *C*.

The above encryption method can be applied to MPEG-I-compressed bit-streams. In this case, the I, P and B frames correspond to the *A*, *B* and *C* blocks, respectively. Hence, all three

types of frames are encrypted but at appropriate security levels. The I-frames receive the highest security. This method causes a tremendous delay, which is intolerable in real-time applications, where multiple encryption/decryption is used. However, the method provides a high level of security.

Batchair et. al. [4] have designed a secured Video-On-Demand (VOD) system in which a service provider transmits video sequences to authorized customers upon the customers' requests. Each customer uses a TV-set top device for communicating with the service provider. To reduce the cost of the set top devices, the researchers use bit-serial encryption techniques rather than block encryption techniques such as DES or RSA. In this system, they use an XOR operation to encrypt the video stream M_i by the encryption key K_i to get the cipher stream E according to the following equation:

$$E(K_i, M_i) = M_i \oplus K_i. \tag{2.1}$$

The original video stream M_i can be recovered at the set-top, by XORing $E(K_i, M_i)$ with K_i. K_i is generated using a Pseudo-Random Shift Register (PRSR). This register has several feedback connections. These connections and the initial contents of the register determine the value of the applied key K_i. PRSR and its polynomial were discussed in Chapter 1 of this thesis. The encryptor, who in this case is the service provider, decides on the feedback connections, which are considered a security element. In this technique, no classification is applied to the data before encryption, and the entire video bit-stream is encrypted. [4]. This technique was not applied to MPEG-I video bit-streams.

Spanos and Maples [6] focus their research on the nature of the MPEG-I bit-stream data. They apply a DES-based technique called Aegis to the I-frames, the MPEG-I video sequence header, and the ISO end-code. The sequence header contains the picture width and height, the bit rate, the frame rate, and the buffer size. It is assumed that encrypting the sequence header and the ISO end-code, the bit-stream will become unrecognizable as an MPEG-I bit-stream. Hence a typical MPEG-I decoder will not be able to decode it. The researchers insert unique stuffing bits, called flag, at both the start and the end of each encrypted section in order to help the receiver identify the encrypted sections of the bit-stream and decrypt it. An iterative method is used to choose the flag to be used. The researchers must make sure that the flag does not appear within the bit-stream. First, a seven-bit flag is chosen, and the bit-stream is examined to determine if it contains a bit sequence, which is the same as the chosen flag. If the sequence of bits which is similar to the flag is detected in the bit-stream, a "0" bit is appended to the flag and the process is repeated until a unique flag is generated. Since this flag generation method is time consuming, it is not suitable for real-time applications. Also, since the flags identify the encrypted portions of the MPEG-I bit-stream, they weaken the security level of the system. A determined cryptanalyst may recognize these stuffing bits and thus solve his synchronization problem. In addition, the sequence header has relatively standard information such as the frame size. So, guessing the right information in the sequence header is not hard for the eavesdropper. Furthermore, since P- and B-frames are not encrypted, some I-macroblocks may exist in these frames. So, if the video has extensive motion, then the P- and the B-frames can reveal a lot of information about the video being encrypted, and Aegis does not provide enough security.

Agi and Gong [5] conduct a number of experiments to test the security of selective encryption of an MPEG-I bit-stream. In their first experiment, they encrypt only I-frames for two video sequences. They observe that the output of the encryption algorithm reveals much about the encrypted video clips. So, they determine the need for a better encryption technique, and suggested increasing the frequency of the I-frames. Although this would improve the security of the system, it would also noticeably degrade the speed. Maximum security can be achieved when all the video frames are encoded as I-frames. In this case, the proposed encryption method is identical to applying encryption to the whole bit-stream, which deviates from the basic idea of selective encryption. Also, this will introduce tremendous compression inefficiency, since the MPEG-I compression algorithm, in this case, ignores the temporal correlation between the frames. In this case, MPEG-I performance is equivalent to Motion JPEG (MJPEG). Agi and Gong [5] suggest the use of a controller to control the I-frame frequency in order to achieve a compromise between system security and compression efficiency. They also conduct two further experiments. In their second experiment they encrypt all I-frames and all P-frames, while in their third experiment they encrypt all I-frames and all I-macroblocks in P-frames. Unexpectedly, they observe no improvement in the security level in either the second or the third over the first experiment. They suspected that this unexpected result is due to the equipment they have used and determined that further study is needed [5].

Li et. al. [18] use the "Pretty Good Privacy" (PGP) crypto-system to encrypt and decrypt MPEG-I bit-streams. Although, PGP was originally designed for encrypting text, they

modify it to deal with MPEG-I frames. First, PGP applies RSA to encrypt the 24-bit key of the International Data Encryption Algorithm (IDEA), then it uses the IDEA for encrypting the video bit-stream data. As do Spanos and Maples [6], the researchers in [18] suggest encrypting only I-frames for general MPEG-I video clips, and all frames (I, P, and B) for highly sensitive applications. They also suggest increasing the frequency of I-frames in the sequence. They test these ideas on a video that has a frame rate of 17.09 frames/second (fps). When they encode all frames as I-frames, this solution degrades the frame rate by 33%, to 11.4 fps [18].

Kunkelmann and Reinema [29] have proposed a method to partially encrypt/decrypt all JPEG- based bit-streams. Their method strongly emphasizes the DCT coefficients. They encrypt only the DC and a few low-frequency coefficients from each block in the compressed digital image (or video). They consider this data as the most important part of each block. Thus, they assume that encrypting these parts from each data block secures the entire bit-stream. However, they report a very low security level. Also, they classify the DCT coefficients rather than the bit-stream in order to apply their method to all JPEG-based standards, such as Motion JPEG (M-JPEG), H.261 and MPEG.

2.3 Thesis experiment set-up

2.3.1 Encryption/Decryption algorithm

As explained in Chapter 1 of this thesis, DES, which is a secret-key encryption/decryption algorithm, has been selected for this research to encrypt/decrypt selected parts of the bit-stream. Furthermore, RSA is used to disguise the DES-key. A complete discussion of securing the key will be presented in Chapter 4 of this thesis.

2.3.2 Stream-parser design

To evaluate different approaches to selective encryption, an MPEG-I bit-stream parser has been implemented in the C programming language. Figure (2.1) shows a functional block diagram of this parser. The parser operates on a full MPEG-I bit-stream. Such a bit-stream consists of multiplexed system, audio, and video streams. In order to extract the video stream, the parser identifies the system header, and then it extracts its packs and identifies the packets within each pack. The audio and the video packets are separated. All video packets that belong to the same picture are assembled together, and they are further processed. The designed video parser then extracts the data to be encrypted and passes it to the encryption sub-system. It does so through control parameters supplied by the user through the C program. After the encryption sub-system encrypts the required portion of the bit-stream, the parser replaces the original data in the video stream with the encrypted data. It then re-packetizes the video stream and re-multiplexes it with the system and audio information and

transmits it or stores it. A circular buffer is implemented to write the processed bits back to where they belong in the bit-stream.

This set-up is used in this work to run various selective encryption/decryption experiments on an encoded MPEG-I bit-stream. However, selective encryption/decryption can be applied during compression, while the MPEG-I encoder is encoding the input digital video. Similarly, at the receiver, the parser processes the received bit-stream and decrypts the encrypted parts of the bit-stream. Then it passes the bit-stream to the decoder. The only considerable delay involved is that caused by the encryption/decryption algorithm.

The above encryption method will produce a video bit-stream that does not conform to the MPEG-I standard format. Thus this seems to be very secure, since most standard MPEG-I players will not be able to identify the bit-stream and play it. Instead they will produce a fatal error and halt the decoding of the bit-stream. Although this bit-stream might seem to be very secure, in reality it is not secure at all. A determined eavesdropper might be able to design his own MPEG-I decoder and attempt to decode as much as possible from the bit-stream to reveal as much information as possible. In fact some commercially available MPEG-I decoders such as the VMPEG 1.7d-lite (by Stefan Eckart), can be used for this purpose. This decoder makes pre-defined decisions when it encounters certain anomalies in the bit-stream, and then continues decoding.

In this thesis, the VMPEG 1.7d-lite decoder will be used to evaluate the security level of selective encryption. However, since the users do not know the pre-defined decisions that this

decoder makes when it attempts to decode an encrypted part, the resulting artifacts in the decoded video can not be fully analyzed. Therefore, an additional method will also be used in this thesis to evaluate and analyze the results. This method exploits the main idea behind encryption in order to allow standard MPEG-I decoders to skip over unknown data and continue decoding. For this purpose, the encryption algorithm is considered to be a black box. For every combination of zeros and ones presented at the input, the box performs certain computations, then outputs another combination of zeros and ones that represents the encrypted data. The input and output data are of the same length, which is 64 bits in the case of the DES algorithm. Therefore, to simulate encryption and force the standard decoder to skip over the encrypted data, the encrypted part is replaced with a string of zeros of the same length as the encrypted data. A standard MPEG-I decoder is designed to skip over zeros in the bit-stream. In this case, the information represented in this part of the bit-stream is lost. The effect of this process on the decoded video depends on the type of lost information.

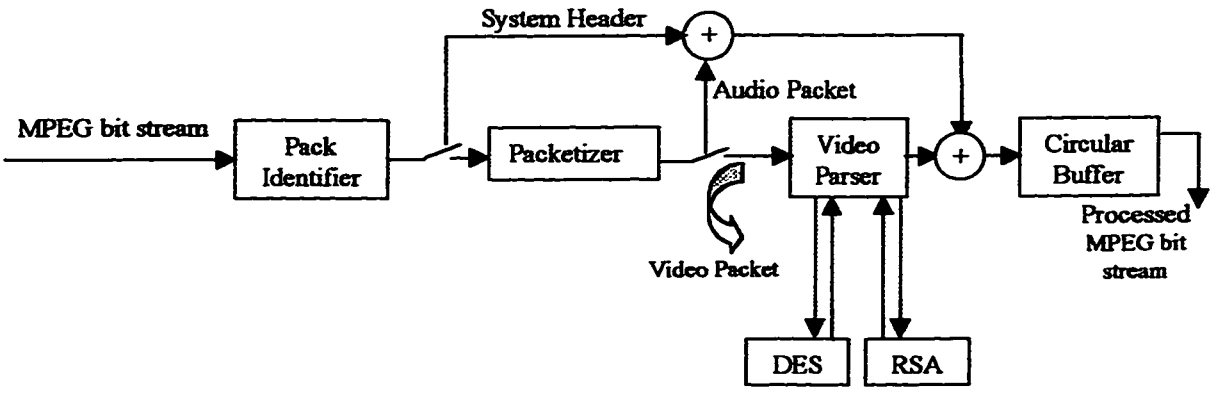


Figure 2.1: Structure of MPEG-I Bit-stream Parser.

2.3.3 Nature of the MPEG-I test video

Eight MPEG-I video clips are used in this thesis for the purpose of testing and evaluating selective encryption methods. These sequences are refereed to as *"Fibon"*, *"Para"*, *"Creamedgates"*, *"Gulf-War"*, *"Kangaroo"*, *"Fish"*, *"Falcon"* and *"OJ-Chase"*, and they are shown in the Appendix. Each of these clips has unique contents and motion characteristics. The *"Fibon"* clip is an educational video about the Fibonacci series. The motion in this clip is slow and there are no hard scene changes (cuts). However, it contains very smooth gradual scene changes. The *"Para"* clip shows paratroopers jumping from an airplane. Thus, it has high motion. The *"Creamedgates"* video clip shows a group of people moving at the entrance to a building. The *"Gulf-War"* clip shows a tank moving in the desert. The motion in this clip is normal; hard scene changes exist in this clip. The *"Kangaroo"* clip shows a kangaroo running in the outback. In this clip, the camera is moving with the kangaroo. The motion of the kangaroo is high but it does not appear in the video clip because the camera moves with the kangaroo. It is interesting to note that the motion in the *"Para"* clip is caused by the paratroopers, while in the *"Kangaroo"* clip, the *background* keeps changing because of the plants that appear in the video, though this change is not very noticeable. The *"OJ-Chase"* clip shows several cars chasing a single car on a highway. The camera is recording the chase from a helicopter. This kind of capturing is called camera tracking. The *"Falcon"* clip shows a falcon catching a fish from a lake.

The first three sequences existed as MPEG-I compressed bit-streams, while the rest of the sequences were compressed in the lab using VITEC MPEG-I Maker Version 1. The

encoding parameters are shown in Table (2.1). The frequency of the I-frame is 1/15 in all video clips, and the coding pattern is IBBPBBPBBPBBPBB. Every picture is encoded as fifteen slices. In addition, the encoder uses default quantizer matrices and full search for the Motion Vectors (MV's).

GOP size	15
GOP pattern	IBBPBBPBBPBBPBB
Quantization	Variable Quantizer Scale; Default Quantizer Matrices;
Number of slices per picture	15

Table 2.1: Encoder parameters.

2.4 Encrypting I-frames

2.4.1 Experiment methodology and interpretation of results

The first experiment conducted in this thesis is encrypting I-frames only. This experiment has been conducted with all test video clips. In order to decode the resulting video bit-stream, an eavesdropper must know the encryption algorithm, the decryption key, and the selective encryption method used with this bit-stream. Although the first and the last information might be relatively easy to guess, the decryption key might be very hard to figure out. Hence, the eavesdropper might decide not to go through cryptanalysis calculations to decrypt the I-frames. Instead, he will ignore the encrypted I-frames and try to get the most from the unencrypted part of the bit-stream. In this case, the most he can get out of the bit-stream can be obtained using the VMPEG 1.7d-lite decoder with the encrypted bit-streams. The same

results can be obtained using a standard MPEG-I decoder with the same bit-streams, but with all the bits representing the I-frames replaced with zeros.

Figure (2.2) shows the results of decoding the first nine pictures of the encrypted (zeroed) *"Para"* video clip. The first picture is an I-picture and hence it is blank. In the P-frames (4, 7,10, and 13) some MB's are predicted from the I-frame and hence they are blank, others are intra-coded MB's (self-contained) and hence they look like normal blocks. The number of these intra-coded MB's increases as the temporal number of the corresponding P-frame increases. This is because the further the P-frame is from the first frame the less correlation it has with that frame. Similarly, in the first two B-frames (2 and 3) some MB's are predicted from the I-frame and some others are predicted from the next P-frame. These MB's are mostly blank, since the I-frame and most of the next P-frame are blank. The other MB's are intra-coded and hence they look normal. In the other B-frames (5, 6, 8, 9, 11, 12, 14, and 15), most MB's are predicted from the nearest P-frames, and the rest are intra-coded MB's. Since the number of normal-looking blocks in the P-frames increases with their temporal number, the number of normal-looking blocks will also increase with the temporal number of the B-frames.

Figure (2.3) shows selected frames from another GOP where a paratrooper's foot, a paratrooper's hand and the wings of the airplane are very clear. Thus, without decrypting the disguised bit-stream, full information can be obtained from the unencrypted part of the bit-stream.

When this experiment is conducted for a video that has a scene change, such as *"Gulf-War"*, the result is even worse. Figure (2.4) shows the frames where a scene change occurs. Figure (2.4-a) is the frame before the scene change and Figure (2.4-b) is the frame after the scene change. This figure shows a complete picture as if no encryption has been applied. This is due to the Intra MB's in P- and B- frames. In each predicted frame, the encoder tries to predict each MB from certain reference pictures by using Motion Compensation (MC) techniques. If the prediction error is too high, the encoder encodes the corresponding MB as a stand alone or self–contained (Intra) MB. If the video clip has high motion, then the number of I-MB's in B- and P-frames becomes higher. In addition, if a scene change occurs at a B- or P-frame, then MC fails to predict the MB's of that frame. Hence, almost all MB's of that frame are encoded as I-MB's. The results for the other video clips are shown in the Appendix.

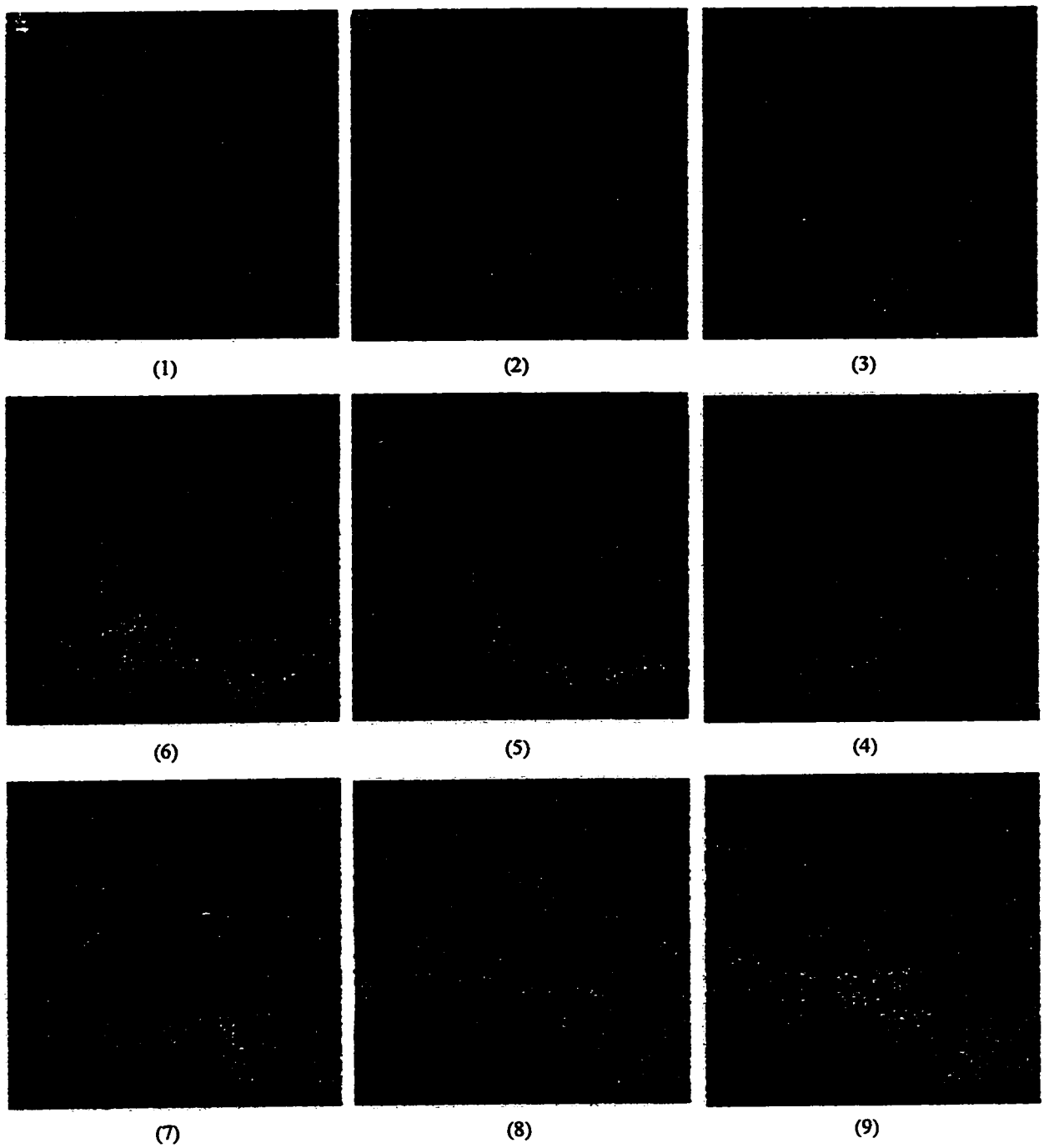
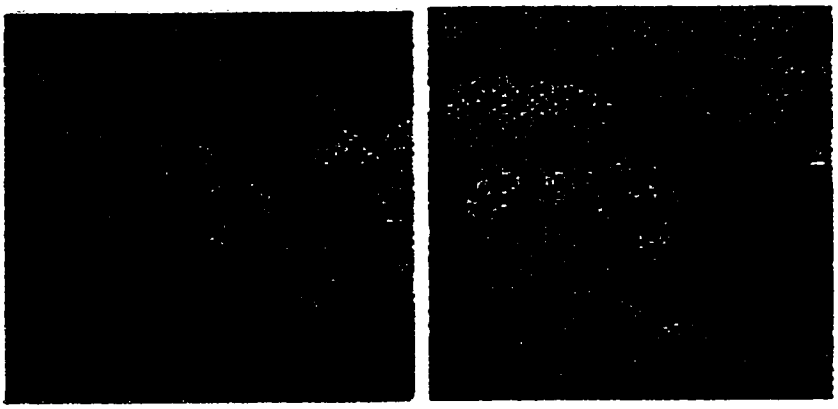
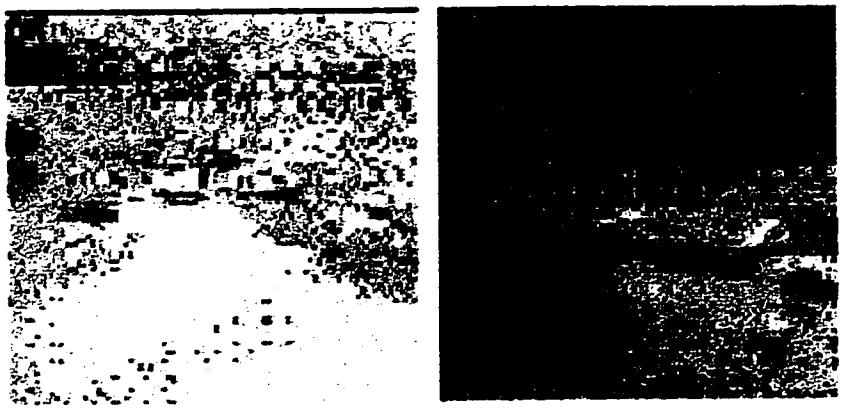


Figure 2.2: The first nine pictures of the I-frames-encrypted *"Para"* video clip. The numbers represent the temporal reference of each picture.



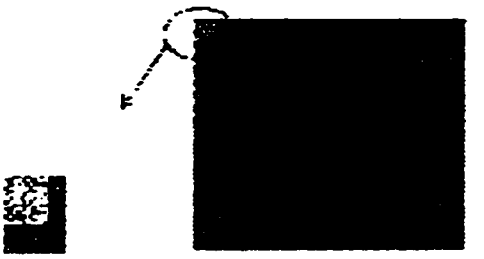
(a) (b)

Figure 2.3: Selected frames from the I-frames-encrypted mode of *"Para"* video clip.



62 63
(a) (b)

Figure 2.4: Two consecutive frames from the encrypted (or zeroed) *"Gulf-War"* video clip at a scene change; (a) is before the scene change and (b) after the scene change.



(a) (b)

Figure 2.5: The zeroed I-frame with the MB that contains a random pattern of colors. (a) is the upper left corner of the image in (b) with the MB that has a random color pattern. (b) is the first encrypted (zeroed) I-frame.

One thing to notice in all the zeroed video clips is the upper left MB, which contains a random pattern of colors as shown in Figure (2.5). This can be explained by examining the following decoding syntax, which is specified by the MPEG-I standard:

```
do{
        macroblock();
} while ( next_bits NOT EQUAL TO "000 0000 0000 0000 0000 0000");
```

The decoder in the above loop tries to decode the first MB without checking for a start code. This is because each slice must contain at least one MB. So, the decoder attempts to interpret the encrypted data for the first MB, which leads to the random looking pattern shown in Figure (2.5). When the decoder finishes decoding the first MB, it finds 23 zeros, and hence it does not try to decode any more MB's. For these MB's the frame buffer is empty and hence the decoder displays blank blocks on the screen. When the decoder tries to decode the second I-frame, it does the same thing again. However, the decoder buffer contains the last P-frame content and hence it displays the last P-frame in place of the current I-frame.

Increasing the frequency of the I-frames was suggested in [5, 18] in order to improve the security level of the I-frame encryption method. For video clips that do not contain hard scene changes, this will reduce the MC error between an encoded P- or B-frame and the reference frames, which in turn will produce a fewer number of I-MB's, which increases the security level. However, this solution does not satisfy the main goal of selective encryption, which is to reduce the processing time of the encryption/decryption process. Furthermore, this solution does not work well for video clips that contain scene changes. This is because increasing the frequency of the I-frames does not guarantee encoding the frame at the scene change as an I-frame.

For video clips, which contain hard scene changes, a possible method for increasing the security level of I-frame encryption is to encode the frame that has a scene change as an I-frame. This requires an algorithm to detect scene changes before compression [30]. If the encryption process works on pre-compressed video, then this requires detecting scene changes in the compressed bit-stream and recompression of the desired images. In this case, this method increases the processing time tremendously.

Another possible improvement in the security level can be obtained by encrypting I- and P-frames [5]. This will increase the security level considerably. For videos with a high number of I-MB's in B-frames, this method will not disguise the MPEG-I video completely. Figure (2.6) shows a sample of pictures of the I-&-P-frames-encrypted *"Gulf-War"* video clip. However, the degradation in the frame rate is tremendous as calculated in the next section. Also, this method deviates from the main idea of selective encryption/decryption. Classifying

P-frames as part of the independent data is nearly invalid because P-frames depend on I-frames and they are not stand-alone data.

(a) (b)

Figure2.6: Sample frames of the I- and P-frames encrypted *"Gulf-War"* video clip.

2.4.2 Processing time reduction

In the previous section, it has been demonstrated that encrypting I-frames only does not provide enough security for highly sensitive and real-time applications. In this section, the reduction in the processing time of this method is calculated. To do this, the number of bits contained by I-frames has to be compared to the total number of bits in all frames. In this thesis the fraction of the video data being encrypted, τ , used to compare different methods of selective encryption/decryption is:

$$\tau = \frac{\text{Number of Encrypted Bits}}{\text{Total Number of Bits in All Frames}} \tag{2.2}$$

As τ becomes smaller the processing time of the proposed method becomes less than that of the thorough encryption method. For thorough encryption, τ is one, which is the maximum value for τ. For the different video clips used in this thesis, the number of bits in the I-frames is listed in Table (2.2). The fraction of the video data being encrypted, τ , for the method of encrypting I-frames is calculated and listed in Table (2.4). From these calculations, the percentage of the data encrypted by classifying I-frames as the independent part varies between 13% and 22%. This gives high efficiency for this selective encryption/decryption scheme.

Video Clip Name	Total Number of Frames	Total number of I-Frames	Total Number of Bits in All Frames	Total Number of Bits in I-Frames
"Fibon"	6773	453	278075520	61158528
"Para"	87	6	3142288	539824
"Creamedgates"	226	15	4505944	879216
"Gulf-War"	324	22	12822376	2446256
"Kangaroo"	149	10	4633552	830264
"Fish"	272	19	8178600	1072216
Falcon	164	11	5855736	1157736
"OJ-Chase"	703	47	22745368	4238576

Table 2.2: Statistics on the test video clips for the data content in the I-frames.

Video Clip Name	Average Number of Bits in one I-Frame	Average Number of Bits in one Non-Intra-Frame
"Fibon"	132953.32	34360.37
"Para"	89970.67	32129.19
"Creamedgates"	58614.4	17188.28
"Gulf-War"	90602.07	28195.98
"Kangaroo"	83026.4	27361.78
"Fish"	56432.42	28088.47
"Falcon"	105248.72	30705.88
"OJ-Chase"	90182.47	28211.57

Table 2.3: Average number of bits in both one I-frame and one non-intra-frame.

Video Clip Name	τ
"Fibon"	0.220
"Para"	0.172
"Creamedgates"	0.195
"Gulf-War"	0.191
Kongharoo	0.179
"Fish"	0.131
"Falcon"	0.198
"OJ-Chase"	0.186

Table 2.4: τ obtained by encrypting I-frames only.

2.5 Processing time of the suggested improvements on the method of encrypting I-frames

The researchers in [18] suggested increasing the I-frame frequency. This will deteriorate the transmission rate. Suppose that the average number of bits in each I-frame is I and in each non-intra-frame is N. Also, let M be the GOP size and let each GOP contains one I-frame every F frames. Also, let the total number of frames in a video clip be K. Hence, τ is calculated as shown in equation (2.3).

$$\tau = \frac{I \times J}{I \times J + N \times (K - J)} \tag{2.3}$$

J represents the number of I-frames that exist in the MPEG video sequence. In the video sequence, there will be an integer number of GOP's and a number of frames that do not compose one complete GOP. Let the number of I-frames in the complete GOP's J_1 and the number of I-frames in the remaining frames J_2. These quantities are given by:

$$J_1 = \left\lceil \frac{M}{F} \right\rceil \times \left\lfloor \frac{K}{M} \right\rfloor \tag{2.4}$$

$$J_2 = \left\lceil \frac{\left\lfloor K - M \times \left\lfloor \frac{K}{M} \right\rfloor \right\rfloor}{F} \right\rceil \tag{2.5}$$

$$\text{Hence, } J = \left\lceil \frac{M}{F} \right\rceil \times \left\lfloor \frac{K}{M} \right\rfloor + \left\lceil \frac{\left\lfloor K - M \times \left\lfloor \frac{K}{M} \right\rfloor \right\rfloor}{F} \right\rceil \tag{2.6}$$

In equation (2.3) above, both N and I depend on the video. Table (2.3) lists the average number of bits in an I-frame for the different video clips. In addition, for different values of F, which determines the I-frame frequency, Table (2.5) has been constructed for the "*Fibon*" clip. As given in the table, the fraction of the video data being encrypted, τ , increases as the I-frame frequency increases. For real-time applications, this degradation is not tolerable. Increasing the I-frame frequency will enhance the quality of the video as discussed in Chapter 1 of this thesis. On the other hand, for a high compression ratio, the GOP must contain as few I-frames as possible [17]. So, a compromise has to take place. However, for security purposes, as supported by Shannon's secrecy theory, high compression adds to the security of the system [11]. Hence, increasing the I-frame frequency does not support these goals and other methods of improving the security level have to be developed. These methods are proposed and discussed in Chapter 3 of this thesis.

Frequency of Intra Frames (F) M = 15	τ
1	1
2	0.816
3	0.659
4	0.585
5	0.492
6	0.492
7	0.492
8	0.373
9	0.373
10	0.373
11	0.373
12	0.373
13	0.373
14	0.373
15	0.217

Table 2.5: The fraction of the video data being encrypted, τ , for different I-frame frequencies for the *"Fibon"* video clip.

For the other seven video clips, the fraction of the video data being encrypted, τ , for different I-frame frequencies is listed in Table (2.6). Equation (2.3), Table (1.2) and Table (2.3) are used to calculate τ in Table (2.6).

Video Clip Name M = 15	**F**	τ
"Para"	3	0.583
	15	0.172
"Creamedgates"	3	0.633
	15	0.206
"Gulf-War"	3	0.617
	15	0.191
Kangharoo	3	0.605
	15	0.179
"Fish"	3	0.503
	15	0.131
"Falcon"	3	0.634
	15	0.198
"OJ-Chase"	3	0.616
	15	0.186

Table 2.6: The fraction of the video data being encrypted, τ , for two I-frame frequencies (F) for seven video clips.

To see how τ behaves as F decreases. Figure (2.7) shows a plot of τ versus F for two different video clips, the *"Fibon"* and the *"Fish"*. These two clips have been chosen because the difference in τ is highest for a fixed F. A similar plot for any other video clip of the ones in Table (2.6) will fall between the two curves in Figure (2.7).

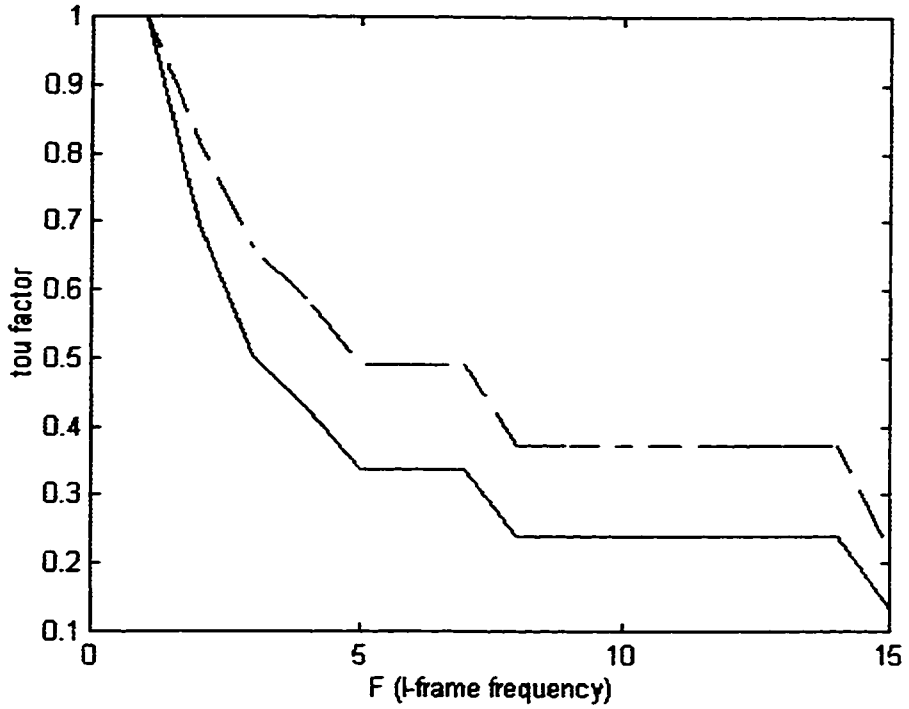


Figure 2.7: A plot of τ versus F for two video clips. "— and -" are for the *"Fibon"* and the *"Fish"* video clips, respectively.

The other improvement suggested by the researchers in [5] is to encrypt P-frames in addition to I-frames. Encrypting I- and P-frames increases the security level as experimented in the previous section. However the degradation in the processing time is expected to be high. To calculate the τ factor for this method, the total data contents of all P-frames for each video clip is listed in Table (2.7). The I-frame frequency is 1/15. Also, the table lists the average data content of all I-frames, which is listed in Table (2.2). The I-frequency is one I-frame in each fifteen frames. Also, the GOP size is fifteen. Table (2.7) shows how encrypting both I- and P-frames degrade the processing time considerably. In addition to this high degradation,

the method does not secure the MPEG-I video completely. Furthermore, classifying P-frames as independent data is not valid because P-frames consist mainly of intra-macroblocks (I-MB) and non-intra-macroblocks. The latter depends completely on I-frames and hence they are dependent data. Hence, this classification of data does not ensure security and does not cut the processing time.

Video Clip Name	Total Number of Bits in all I-Frames	Total Number of Bits in all P-Frames	τ
"Fibon"	61158528	100513792	0.581
"Para"	539824	1257328	0.572
"Creamedgates"	879216	2229896	0.690
"Gulf-War"	2446256	6751088	0.717
Kongharoo	830264	2714384	0.765
"Fish"	1072216	6316360	0.903
Falcon	1157736	3024288	0.714
"OJ-Chase"	830264	2714384	0.765

Table 2.7: The fraction of the video data being encrypted, τ , resulting from encrypting all I- and P-frames.

2.6 Summary

The main conclusion from these experiments is that encrypting I-frames only does not give a satisfactory level of security, although it reduces the required processing time considerably.

Many suggestions to improve the level of security have been evaluated. These suggestions include increasing the I-frame frequency [5, 18] and encrypting both I- and P-frames [5] in

the video clip. Although, the security level is improved by these solutions, they diverge from the main idea of selective encryption. Also, classifying P-frames as part of the independent data is not valid because they depend on I-frames, as known from the structure of MPEG-I-compressed bit-stream as discussed in Chapter 1. In addition, τ resulting from the method of increasing I-frame frequency becomes more than 0.50 without disguising the video completely. In the case of encrypting both I- and P-frames, τ becomes very close to 1, which diverges from the purpose of selective encryption. Hence, improvements have to be made according to the idea of selective encryption. These methods are presented and fully analyzed in the next chapter.

Chapter 3

Improving the Security Level of Selectively-Encrypted MPEG-I Video Bit-streams

3.1 Introduction

The calculations of Chapter 2 showed that selective encryption reduces the processing time considerably. It also showed that encrypting only I-frames in an MPEG-I video bit-stream does not provide the required security level for some applications. Very sensitive applications such as military require the content of the transmitted material to look like noise whenever an eavesdropper attempts to decode the video. However, this is not the case when only I-frames

are encrypted. Typical video sequences have many scene changes, hence many frames of the encrypted video will be successfully decoded (without decryption) to reveal part of the sensitive contents. On the other hand, the achieved security level may be adequate for commercial TV broadcast, since the quality of the decoded video (without decryption) is completely unacceptable to the viewer. The viewer is not interested in watching the storyboards of a movie buried within a scrambled video sequence as much as he is interested in enjoying the full movie.

In this chapter two new selective encryption methods will be developed to achieve better security level. The first is stated in section 3.2 and investigated in Section 3.3, while the other method is stated in section 3.4 and investigated in Section 3.5. Section 3.6 presents further improvements of the methods suggested in sections 3.2-3.5. Section 3.7 discusses a new scheme that gives the minimum processing time. Section 3.8 is a summary of the chapter.

3.2 The first proposed classification

The resulting MPEG-I video clips, from the conducted experiments in chapter 2, showed that as the temporal number of the frames within the GOP increases (move away from the first picture in the group, which is intra (I)), the contents of the frames become more obvious. This is due to the existence of intra-encoded MB's in P- and B-frames. These I-MB's are encoded as such when the Motion Compensation (MC) technique fails to predict the MB from the reference frames. As the frame moves away from the I-frame, its content changes, hence the number of I-MB's increases until the next I-frame.

These results suggest encrypting the intra-coded MB's in the P- and B-frames in addition to encrypting the I-frames, in order to improve the security level of the selectively-encrypted MPEG-I video bit-stream. This means that all I-MB's in all frames must be encrypted. Hence, all I-MB's in all frames constitute the independent part of the MPEG-I video bit-stream. This will improve the security level, especially at the scene changes without requiring scene change detection during compression as explained in Chapter 2 of this thesis. Section 3.3 discusses this classification thoroughly.

3.3 Encrypting all I-macroblocks in all frames

3.3.1 Experimental methodology and results

The same experiment set-up explained in Chapter 2 is used here. However, the parser is made to parse then pass all I-MB's in all frames to the DES algorithm to be encrypted/decrypted. The type of the MB is identified by a variable length code (VLC) in the MB header. Hence, the parser first decodes the header of each macroblock in order to identify its type. In MPEG-I, each frame type has a set of allowed MB_type. These types are represented by a set of code words. Table (3.1) shows the MB_type code words and their meanings for P- and B-pictures [17]. In addition to other information, each code word in the table indicates whether the MB is intra or non-intra. In the table, if the flag *I* has a value of 1, then the MB that has the corresponding VLC of the MB_type is intra and vice versa. So, the parser checks the MB type and accordingly it either passes the MB to be encrypted or passes the MB as it is without

encryption. The other flags indicate the motion vectors and the quantizer scale. These flags will be needed and discussed later in the chapter.

The VMPEG 1.7d-Lite and the zeroing method discussed in Chapter 2 for visually evaluating the encrypted sequences are also used here for evaluating the security level of encrypting all I-MB's in all frames. To explain the zeroing method in this case, the decoding procedure of the blocks in each macroblock is presented first.

At the beginning of each block in an I-MB, the decoder reads a VLC that represents either the DCT_DC_SIZE_luminance or the DCT_DC_SIZE_chrominance. The DCT_DC_SIZE_luminance is read if the block is of luminance type, and the DCT_DC_SIZE_chrominance is read if the block is of chrominance type. The DCT_DC_SIZE VLC codes with their information about the size are given in Table (3.2) for both the luminance and the chrominance blocks. This VLC code tells the decoder how many bits were used to encode the DC-DCT coefficient (DC coefficient of the Discrete Cosine Transform). For example, if the DCT_DC_SIZE_luminance codeword is "100", then the DCT_DC_differential for the corresponding block does not exist in the bit-stream. Similarly, for example, if the block is of chrominance type and the DCT_DC_SIZE_chrominance code is "00", then the DCT_DC_differential for the corresponding block does not exist in the bit-stream. As another example, assume that the DCT_DC-SIZE_chrominance code word is "110", then according to Table (2.6-b) three bits are used to encode the DC coefficient of the DCT transform of the block.

MB_type VLC code	Type			
	I	B	F	Q
1	0	0	1	0
01	0	0	0	0
001	0	0	1	0
00011	1	0	0	0
00010	0	0	1	1
00001	0	0	0	1
000001	1	0	0	1

(a)

MB_type VLC code	Type			
	I	B	F	Q
10	0	1	1	0
11	0	1	1	0
010	0	1	0	0
011	0	1	0	0
0010	0	0	1	0
0011	0	0	1	0
00011	1	0	0	0
00010	0	1	1	1
000011	0	0	1	1
000010	0	1	0	1
000001	1	0	0	1

(b)

Flag	Value	Meaning
I	1	MB is intra
	0	MB is non-intra
B	1	Backward motion vector exists
	0	Backward motion vector does not exist
F	1	Forward motion vector exists
	0	Forward motion vector does not exist
Q	1	Quantizer scale codeword exists
	0	Quantizer scale codeword does not exist

(c)

Table 3.1: Variable Length Codes (VLC's) for the macroblock_type code word that exists in the macroblock header for (a) P-Pictures (b) B-Pictures. (c) Gives the interpretation of the flags (I, B, F, Q) listed in tables (a and b).

VLC code	DCT_DC_SIZE_ luminance
100	0
00	1
01	2
101	3
110	4
1110	5
11110	6
111110	7
1111110	8

(a)

VLC code	DCT_DC_SIZE_ chrominance
00	0
01	1
10	2
110	3
1110	4
11110	5
111110	6
1111110	7
11111110	8

(b)

Table 3.2: Variable Length Codes (VLC) for (a) the DCT_DC_SIZE_luminance code word and (b) the DCT_DC_SIZE_chrominance code word.

In an I-MB, after decoding these two VLC codewords (the DCT_DC_SIZE_luminance/ chrominance and the DCT_DC_differential), the decoder starts decoding the DCT_COEFF_NEXT codewords. Each codeword gives a pair of (run, length) that is used in decoding the zig-zag scanned quantized AC-DCT coefficients (AC coefficients of the Discrete Cosine Transform). The decoder keeps decoding these VLC codes until it reaches the End of Block (EOB) code, which is "10". Next, the decoder starts decoding the next block and so on.

To simulate the effect of encrypting all I-MB's (i.e. zeroing) the DCT_DC_SIZE_ chrominance is made "100" in each chrominance block in each I-MB. Similarly, DCT_DC_SIZE_luminance is made "00" in each luminance block in each I-MB. This informs the decoder that there is no codeword for the DC-DCT coefficient and the decoder will not look for the DCT_DC_differential but instead it will decode the DCT_COEFF_NEXT codewords. However, after inserting zero-size codewords, the End of Block (EOB) code "10" is inserted to indicate the end of the block. All the remaining AC coefficients are removed from the bit-stream. However, to keep the integrity of the bit-stream, the number of bits removed from each block is calculated and an equivalent number of zeros are appended at the end of the corresponding slice. As a result, the decoder reads nothing for the DCT coefficients or equivalently the blocks in all I-MB's have no data. It will also skip all the zero bits appended at the end of each slice. This modification process (zeroing) is clarified in Figure (3.1) with an example where a slice contains two I-MB's only. The shadowed part represents the bit-stream of the I-MB. Part (a) of the figure shows the slice as it is received by the parser. The bit-stream of each macroblock contains a header and six blocks (the first our blocks are luminance and the other two blocks are chrominance). Each block contains the DC coefficient of the DCT transform followed by the AC coefficients. Each block ends with the End-of-Block (EOB) code "10". The shadowed part in Figure (3.1-a) is the bit-stream in the slice. The parser does the required replacements explained above to emulate the encryption through zeroing to produce the slice shown in part (b) of the figure. The parser keeps the header of each I-MB as it is. However, at the beginning of each block, it inserts a code that indicates the non-existence of the DC-coefficient of the

DCT transform of that block. This code is "00" in case of chrominance block and "100" in case of luminance block. Then the parser removes an equivalent number of "0" bits at the end of the slice to keep the integrity of the bit-stream as indicated by the block of zeros at the end of the slice in Figure (3.1-b). This method is just to evaluate visually the proposed encryption scheme using standard MPEG-I decoder.

When all the I-MB's in all frames of the test video clips are encrypted as discussed before, it was noticed that the security level has increased considerably. Figure (3.2) shows the frames 23, 34 and 40 of the *"Para"* video clip after decoding the simulated encryption using the zeroing method. The set in part (a) are obtained by zeroing I-frames only while set (b) is obtained by zeroing all I-MB's in all frames. From these frames, the improvement in the security level is clear as have been verified by several colleagues.

For video clips, which have scene changes, encrypting I-MB's enhances the security level considerably. If the scene change occurs at a non-intra frame, then most MB's of that frame are intra. While encrypting I-frames alone does not secure such video clips, encrypting I-MB's in all frames provides a security level for such clips. Figure (3.3) shows the frame before (frame number 62) and the frame after (frame number 63) a scene change in the *"Gulf-War"* video clip. (a) and (b) are before the frames before a scene change and the frame number is (62). (c) and (d) are the frames after the scene change and the frame number is (63). These frames are obtained using a standard MPEG decoder with the zeroed MPEG sequence. The set at the left (a and c) is from the only-I-frames-encrypted version, while the right set (b and d) is from the I-MB's-encrypted version of the video clip. Other results for

other video clips are shown in the Appendix. These results have been tested by several colleagues and the results are similar to those obtained for *"Gulf-War"* and *"Para"* video clips.

A slice containing two I-MB's

	I-MB-1							I-MB-2						
	B1	B2	B3	B4	B5	B6		B1	B2	B3	B4	B5	B6
H1	dc-L dct1 . . . EOB	dc-L dct1 . . . EOB	dc-L dct1 . . . EOB	dc-L dct1 . . . EOB	dc-C dct1 . . . EOB	dc-C dct1 . . . EOB	H2	dc-L dct1 . . . EOB	dc-L dct1 . . . EOB	dc-L dct1 . . . EOB	dc-L dct1 . . . EOB	dc-C dct1 . . . EOB	dc-C dct1 . . . EOB

I-MB-1 and I-MB-2: The first and the second macroblocks in the slice, respectively.
H1: the header of the first MB. H2: the header of the second MB.
B1: the firs Block in the MB. B2: the second Block in the MB... etc.
dc-L: DC-DCT coefficient of the Luminance Block
dc-C: DC-DCT coefficient of the Chrominance Block
dct1: the first AC-DCT coefficient.
EOB: End of Block code "10".
The shadowed part constitutes the bit-stream of the slice.

(a)

The slice contents after emulate encryption through zeroing

	I-MB-1							I-MB-2						
	B1	B2	B3	B4	B5	B6		B1	B2	B3	B4	B5	B6	0000 0000 000..
H1	00 10	00 10	00 10	00 10	100 10	100 10	H2	00 10	00 10	00 10	00 10	100 10	100 10	.....

Each block contains a code word indicating non-existing DC-DCT coefficient; "00" in the first four blocks (Luminance) of each MB and "100" in the remaining two blocks (Chrominance). The "10" code word is the End of Block code. The number of zeros at the end of the slide equal the number of bit in the DCT-coefficients eliminated from the two MB's.

(b)

Figure 3.1: Illustration the process of zeroing the I-MB's. (a) The original slice containing two I-MB's only. (b) The slice after the parser replaces the required code words to implement zeroed I-MB's.

From these results, we conclude that encrypting I-MB's hides the content of each frame in the video clip. However, when these frames are played in motion, it was noticed that the motion could be clearly recognized. In the case of the *"Para"* video clip, the airplane and the body of a moving human being can be easily recognized but the details of the airplane and the body are not clear. The same thing is true for the *"Kangaroo"* video clip, where the motion of an animal can be recognized but the type of the animal and its details as well as the background (the desert) can not be recognized. In the case of the *"Gulf-War"* video clip, a motion of a vehicle is recognized but no more details can be obtained from the encrypted video clip. The most one can tell is that the vehicle is a tank, but he can not recognize its kind.

The second scheme of visually evaluating the encrypted MPEG bit-stream, which resulted from encrypting all I-MB's, is to decode them by VMPEG-1.7d-Lite.

The result of decoding the encrypted MPEG video bit-stream (all I-MB's are encrypted) using VMPEG-1.7d-Lite is shown in Figure (3.4) and Figure (3.5). Figure (3.4) indicates higher security level compared to that obtained when the zeroing scheme was used. For example, in the *"Para"* video clip, the zeroing scheme produces a video where the plane and the paratroopers are easily identified but the details are not. However, the second VMPEG 1.7d-Lite decoding seams to disguise the plane and the paratroopers' bodies but the motion is still very clear when the sequence is played. Figure (3.5) shows two frames (frames number 79 and 216) from the encrypted then decoded *"Fibon"* video clip. The number under each picture indicates its frame number. The figure also shows the corresponding unencrypted frames. The words, the numbers, and the high frequency represented in the edges of the

person's face are easily identified. Furthermore, the motion is very clear when the sequence was played. The motion was also clear in all other test clips. For example a moving body can be easily identified in the "*Gulf-War*" video clip, and a flying falcon can be detected in the "*Falcon*" video clip. The results are shown in the Appendix.

So, for those video clips, which have motion, the method of encryption all I-MB's does not provide enough security level. However, for those videos where the motion is limited, encrypting all I-MB's in all frames provide enough security as in the *"Fish"* video clip.

In order for the decryptor to differentiate between the encrypted I-MB's and other MB's, the MB_type code word is not encrypted. Hence, the encrypted part of the I-MB starts after this code word in the macroblock header. This exclusion does not affect the security level of the encrypted video clip because most of the data of the macroblock is encrypted. Furthermore, this exclusion enables the decryptor to identify the I-MB's. This is clarified in Figure (3.6). Section 3.3.2 calculates the reduction in the processing time achieved by this method compared to thorough encryption of the MPEG-I video bit-stream.

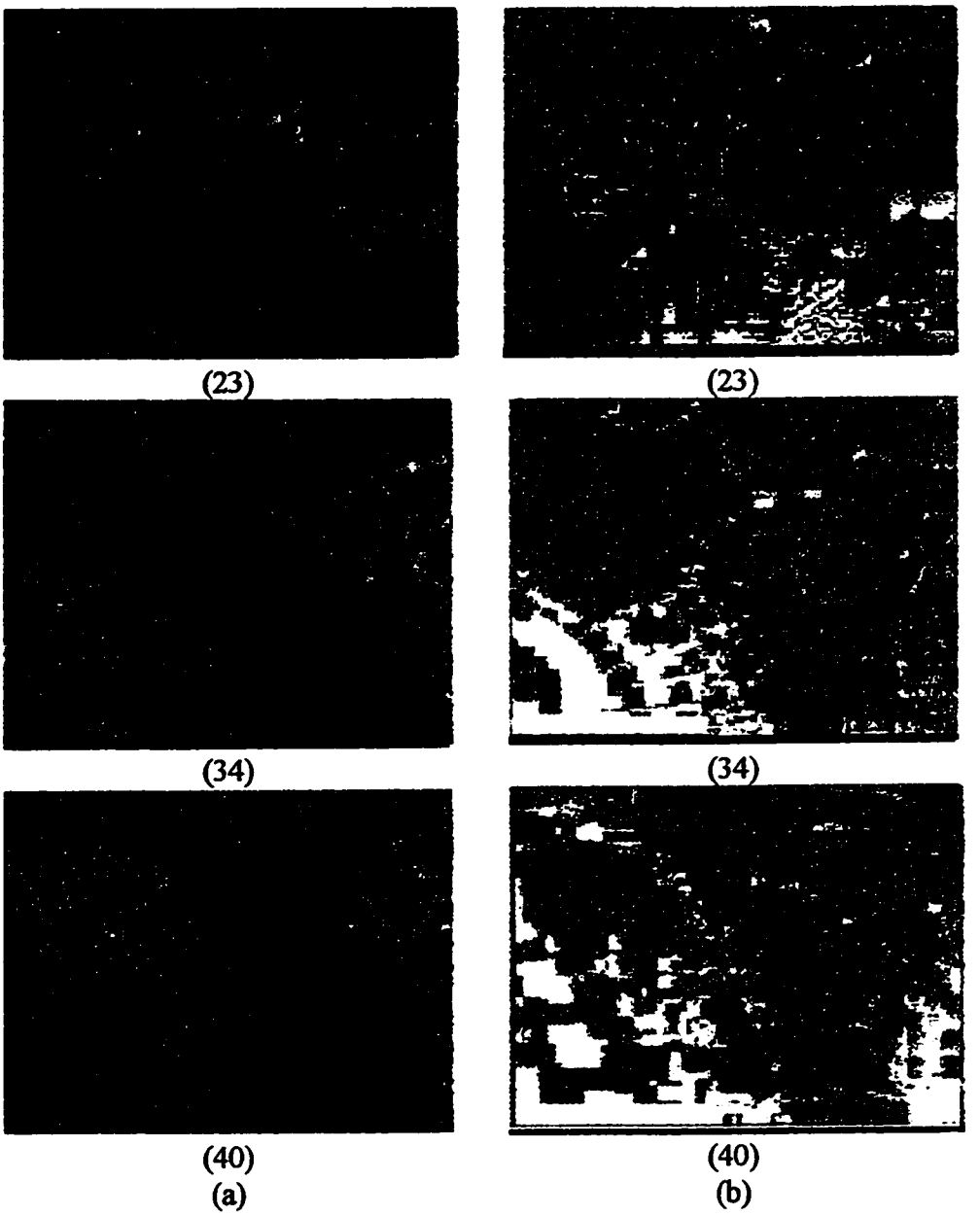


Figure 3.2: Selective Encryption of the *"Para"* video clip. (a) after zeroing all I-frames only. (b) after zeroing all I-MB's in all frames. The numbers are the frame number in the video clip.

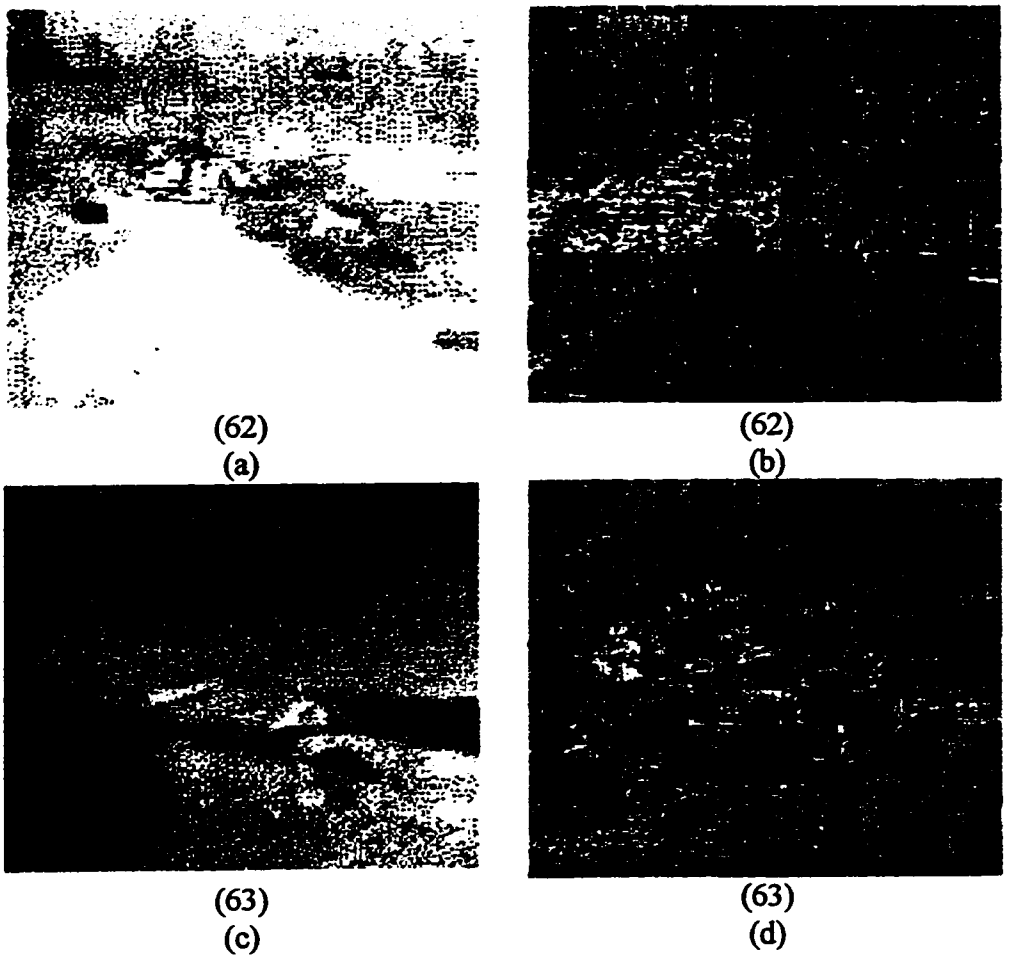


Figure3.3: Selective Encryption of the "*Gul-War*" video clip after decoding using standard MPEG-I decoder. (a) and (c) after zeroing I-frames only (b) and (d) after zeroing I-MB's in all frames.

Figure 3.4: A Frame of the *"Para"* video clip resulted by decoding the real encrypted sequence with the VMPEG 1.7d-Lite decoder.

It was possible to perceive the motion in the decrypted test clips because the motion information was not secure. This motion information is contained in the motion vectors of the non-intra MB's. The MC techniques predict these MB's from the reference pictures. Although the content of the reference pictures is encrypted, the motion vectors themselves are not. This makes the motion of the objects in the picture very clear in the resulting MPEG video clips. In order to disguise the video completely, the motion vectors have to be also encrypted. The method, which achieves this, is discussed in sections 3.4 and 3.5.

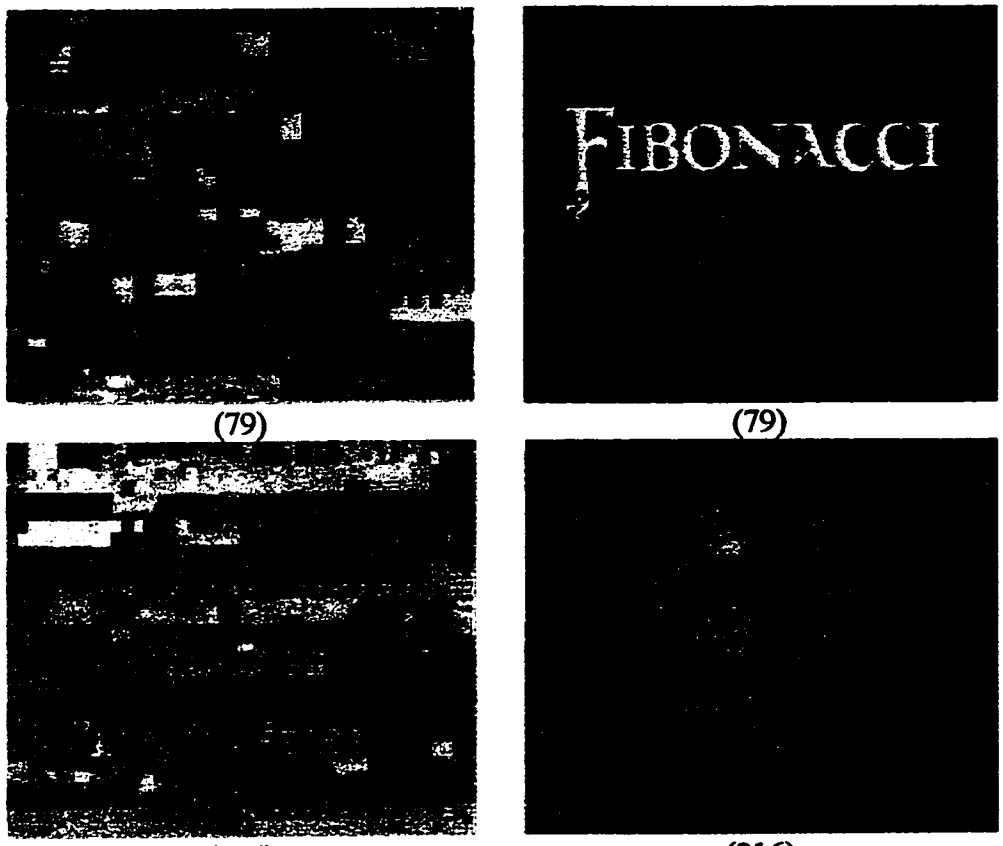


Figure 3.5: Frames from *"Fibon"* video clip. (a) Two frames from the I-MB's-encrypted video decoded by VMPEG 1.7d-Lite. (b) Two frames from the original video clip for comparison.

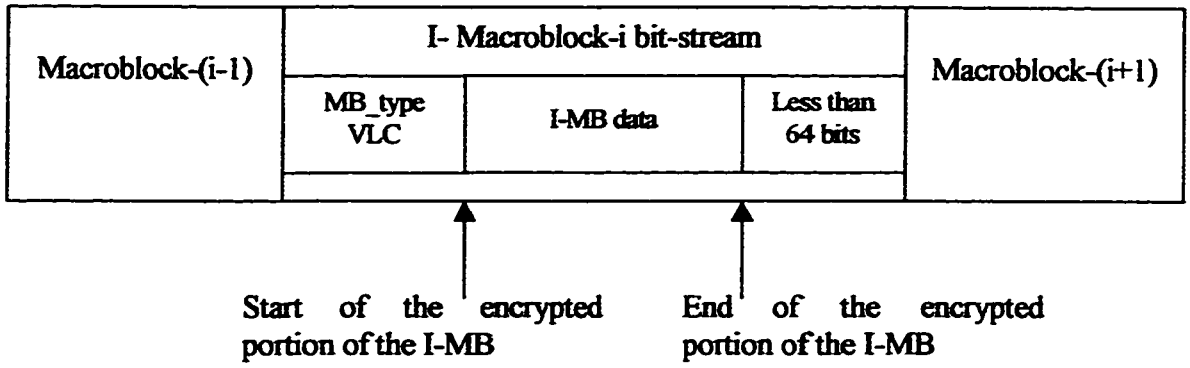


Figure 3.6: The start and the end limits of the encrypted portion of the I-MB in the method of encrypting all I-MB's in all frames.

3.3.2 Processing time calculations

In section 3.3.1, it was demonstrated that encrypting all I-MB's in all frames improves the security level over encrypting I-frames only. However, this may increase the processing time. The amount of increase in the processing time depends on the nature of the video clip. The processing time of encrypting all I-MB's is related to the total number of bits in these blocks. Table (3.3) lists number of MB's, number of I-MB's, number of bits in all MB's, and number of bits in all I-MB's in each video clip used in this thesis. Table (3.4) lists the percentage of number of I-MB's to the total number of MB's. It also lists the percentage of bits in these I-MB's to that contained in all MB's (sequence, GOP, and picture headers are not included).

The processing time of encrypting all I-MB's can be compared to that of the thorough encryption of the entire bit-stream by calculating the fraction of video data being encrypted, τ , which is the ratio of the number of the encrypted bits to the total number of bits in all frames. The second column of Table (3.5) gives τ when only I-frames are encrypted,

and the third column gives τ when all I-MB's in all frames are encrypted. To illustrate the saving this method achieves compared to thorough encryption, suppose that thorough encryption of an MPEG video clip takes one hour. By encrypting all I-MB's in all frames the processing time could be between 14.3 and 20.9 minutes depending on the nature of the video clip. The nature of the video clip affects the performance of the MC techniques. As a result, they affect the number of I-MB's in non-intra frames.

ideo Clip Name	Total Number of MB's	Total number of I-MB's	Total Number of Bits in All MB's	Total Number of Bits in I-MB's
"Fibon"	2213689	308904	269629701	90667842
"Para"	28512	5398	3058259	990381
"Creamedgates"	60528	8144	4353263	1378766
"Gulf-War"	44035	8427	12597550	3098121
"Kangaroo"	12072	4322	4548896	1371729
"Fish"	19879	6741	8023787	2071041
"Falcon"	13294	5882	5762230	2041790
"OJ-Chase"	146130	23688	22110999	5428411

Table 3.3: Statistics on the test video clips for the data content of the I-MB's.

Video Clip Name	% of Number of I-MB's to The Number of All MB's	% of Number of Bits in I-MB's to The Number of Bits in All MB's
"Fibon"	13.954	33.627
"Para"	18.932	32.384
"Creamedgates"	13.455	31.672
"Gulf-War"	19.137	24.593
"Kangaroo"	35.801	30.155
"Fish"	33.910	25.811
"Falcon"	44.246	35.434
"OJ-Chase"	16.210	24.551

Table 3.4: Percentage of the data contained by I-MB's.

Video Clip Name	τ by encrypting I-frames only	τ by encrypting I-MB's in All Frames	The difference in τ
"Fibon"	0.220	0.326	0.106
"Para"	0.172	0.315	0.143
"Creamedgates"	0.195	0.306	0.111
"Gulf-War"	0.191	0.242	0.051
"Kangaroo"	0.179	0.296	0.117
"Fish"	0.131	0.253	0.122
"Falcon"	0.198	0.349	0.151
"OJ-Chase"	0.186	0.239	0.053

Table 3.5: The fraction of video data being encrypted, τ , for the two methods of selective encryption: encrypting only I-frames and encrypting all I-MB's in all frames.

The effect of encrypting all I-MB's can be analyzed by comparing τ for encrypting all I-MB's and that for encrypting all I-frames in each of the test video clips. The difference between both τ 's is calculated and listed in the last column of Table (3.5). This difference varies between 0.053 and 0.151. This variation can be interpreted by examining number of I-MB's that exist in P- and B-frames and number of bits within these I-MB's. These statistics are listed in Table (3.6), which shows that if most of the I-MB's bits are contained within the I-MB's in the P- or B-frames, then τ will be much higher than that for the method of encrypting I-frames only. The *"Kangaroo"* and the *"Fish"* video clips are examples of video with high number of bits in the I-MB's within the non I-frames. On the other hand, the I-MB's, in the *"Gulf-War"* video clip, that are in P- and B-frames contain 25% of the data in all I-MB's and hence the difference in τ is just 0.05. Generally, we can state that if less than 50% of the data content of the I-MB's exist outside I-frames, then the degradation in the processing time caused by further encrypting I-MB's in P- and B-frames in addition to I-frames, is small and acceptable. Meanwhile, encrypting all MB's leads to higher security level.

Video Clip Name	% of Number of I-MB's in P- and B-frames to the number of all I-MB's	% of Number of Bits in I-MB's that are in P- and B-Frames to the Total Number of Bits in All I-MB's
"Fibon"	50.859	33.174
"Para"	63.320	48.308
"Creamedgates"	39.219	36.982
"Gulf-War"	42.328	21.532
"Kangaroo"	61.129	39.880
"Fish"	52.648	48.746
"Falcon"	68.582	43.599
"OJ-Chase"	40.476	22.695

Table 3.6: Statistics on the number of bits in those I-MB's that are in P- and B-frames.

3.3.3 Recognizing the end of the encrypted portion of the I-MB by the receiver

An important issue in selective encryption of I-MB's is how the receiver can recognize the end of the encrypted portion. The encryptor encrypts the data representing an I-MB into segments of 64 bits each. If the last segment does not contain 64 bits, the encryptor does not encrypt it. This is illustrated in Figure (3.6). As explained in section 3.3.1, the receiver starts decoding the I-MB by decrypting the first 64-bit segment, which starts directly after the MB_type code word. Then the receiver decrypt the next 64 bits, and so on, until it reaches the next I-MB or the last segment has less than 64 bits.

The process which the receiver has to perform in order to identify the end of the encrypted portion of the I-MB. After decrypting and decoding a 64-bit segment, the receiver has to perform the Main Decision Block (MDB) shown in dashed box in Figure (3.7) and decides either to decrypt the next 64-bit segment or to decode it. The MDB process can be explained as follows. At the end of each 64-bit segment, there will always be a few bits that are part of a code word representing a DCT-coefficient. The other part of the code word will be at the beginning of the next 64-bit segment. Hence, the receiver concatenates these remaining few bits with the first bits in the very beginning of the next 64-bit segment. If the next 64-bit segment is encrypted, the receiver will not be able to find a valid code word and hence it will know that the next 64-bit segment is encrypted. However if it finds a valid code word, the decoder will know that the next segment is more likely unencrypted. So, it will attempt decoding another code word. This is so because it is unlikely that the DES algorithm will produce two consecutive valid code words in the right positions in the MPEG video bit-stream.

This process introduces some delay at the receiver. To calculate the average of this delay the average number of times the receiver has to decode a code word at the end of each 64-bit segment in an I-MB has to be calculated. The number of valid VLCs exist in the data of any macroblock with their lengths are listed in Table (3.7) according to [17].

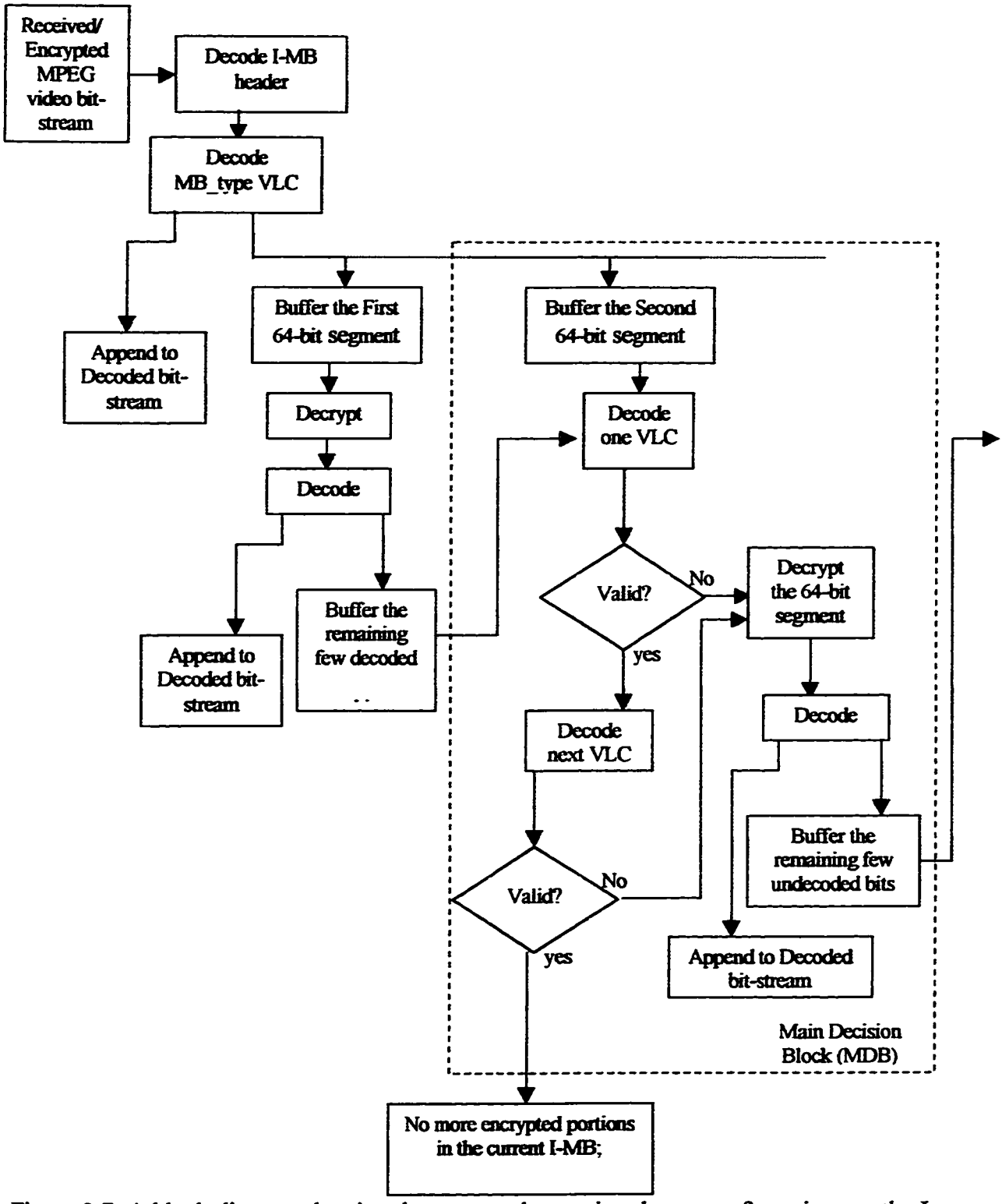


Figure 3.7: A block diagram showing the process the receiver has to perform, in case the I-MB's are encrypted, to determine the end of the encrypted portion of the I-MB.

Length of a Valid VLC	Number of VLCs with this length
2	3
3	2
4	2
5	4
6	6+1(escape code)
7	8
8	8+255(escape codes)
9	16
11	16
12	257(escape codes)
13	30
14	32
15	32
16	32
17	32

Table 3.7: A list of the lengths of the valid MB-Data code words with the number of those code words that have the corresponding length.

According to Table (3.7), the average number of times the receiver has to search a valid codeword at the end of each decrypted 64-bit segment is:

$$\sum \frac{(\textit{length of valid VLC}) \times (\textit{No. of VLCs with this length})}{2} = 4010 \textit{ times} \qquad (3.1)$$

Hence, the average delay caused by this operation at the receiver is:

$$\text{Average delay in the whole video clip} = 4010 \times L \times M \times t_{VLC} \textit{ seconds} \qquad (3.2)$$

Where:

L : *Average number of 64 - bit segments in one I - MB,*

M : *Total number of I - MB' s in the video clip, and*

t_{VLC} : *Time (in seconds) required to decode (search for) one VLC.*

The average delay for a video clip can be calculated in terms of t_{VLC}. To do so for the test video clips used in this thesis, Table (3.8) lists both *L and M* for each video clip. Also, the table calculates the delay in terms of t_{VLC}. To have an idea of the percentage of this delay to the video clip playback time let us assume t_{VLC} to be 3.33 *nsec* (assuming that decoding one VLC requires only one instruction using a 300MHz machine). t_{VLC} depends on the decoder speed and the speed assumed here is just to give an idea of introduced delay. Table (3.9) lists the average delay for each video clip and the percentage of this delay to the playback time of the video. From the table the percentage of the average delay to the playback time is less than 8%. This number is tolerable when improving the security level is very important and using faster machine will make this delay very tolerable.

It is important to mention that this process performed by the receiver affects MPEG-I bit-stream syntax. The Main Decision Block (MDB) shown in Figure (3.7) has to be implemented in the decoder. As previously mentioned in Chapter 1, MPEG standard does not specify the encoder but it specifies the decoder. Hence, the decryptor must have a functionality to perform the MDB. This introduces slight change in the decoder.

Video Clip Name	L	M	The Average Delay
"Fibon"	4	308904	$4.9\times10^{9}\times t_{VLC}$
"Para"	2	5398	$43.3\times10^{6}\times t_{VLC}$
"Creamedgates"	2	8144	$65.3\times10^{6}\times t_{VLC}$
"Gulf-War"	5	8427	$16.9\times10^{7}\times t_{VLC}$
"Kangaroo"	4	4322	$69.3\times10^{6}\times t_{VLC}$
"Fish"	4	6741	$10.8\times10^{7}\times t_{VLC}$
"Falcon"	5	5882	$11.8\times10^{7}\times t_{VLC}$
"OJ-Chase"	3	23688	$28.5\times10^{7}\times t_{VLC}$

Table 3.8: The average delay caused by the process illustrated in Figure (3.7) for the test video clips used in this thesis.

Video Clip Name	The Average Delay (second)	Playback time of the video clip (seconds)	Percentage (%) of the average delay to the playback time
"Fibon"	16.30	226	7.21
"Para"	0.144	3	4.80
"Creamedgates"	0.218	9	2.42
"Gulf-War"	0.563	13	4.33
"Kangaroo"	0.231	4	5.78
"Fish"	0.360	9	4.00
"Falcon"	0.393	5	7.86
"OJ-Chase"	0.950	23	4.13

Table 3.9: The average delay caused by the process illustrated in Figure (3.7) assuming t_{VLC} is 3.33 *nsec* for the test video clips used in this thesis.

3.4 The second proposed classification

From the results discussed in section 3.3, it was concluded that the main drawback of encrypting all I-macroblocks in all frames is the visibility of the motion. Hence, in order to completely disguise an MPEG video sequence, one must hide the motion vectors of all non-intra coded MB's of the P- and B-frames. As explained in Chapter 1 of this thesis, these motion vectors (MV) are contained within the headers of the non-intra coded MB's. This suggests encrypting the headers of P- and the B-MB's in addition to encrypting all the data in all I-MB's in the sequence. Hence, the new classification of MPEG-I video bit-stream is to consider both all I-MB's in all frames and the headers of all non-intra macroblocks as the independent part of the bit-stream.

3.5 Further encrypting the headers of P- and B-macroblocks

3.5.1 Experimental methodology and results

The same parser discussed before is used here for encrypting the headers of the P- and B-MB's and all I-MB's. The parser passes all I-MB's and the headers of the non-intra macroblocks to the encryption/decryption sub-system (DES algorithm) to be encrypted. Since the input block for the DES algorithm is 64-bit long, the parser must pass 64-bit segment of each header to the DES for encryption. However, the header might be less than 64-bit long. So, there are two choices to perform this method. The first choice is to group the headers of the P- and the B-macroblocks of one slice together into a header sub-bit-stream and the data

of the P- and B-macroblocks into data sub-bit-stream. Then, each 64-bit segment of the header sub-bit-stream is passed to the DES algorithm for encryption. The second choice is to pass the first 64 bits of each non-intra MB's to the DES algorithm. This data may contain the entire header and a small part of the data in the corresponding non-intra MB. Figure (3.8) clarifies these two schemes with a specific example where a slice has five non-intra macroblocks. (a) The original slice (before encryption) having five macroblocks. (b) The first encryption scheme suggesting encrypting the header of each macroblock and some data of the macroblock to constitutes a 64-bit segment. (c) The second encryption scheme suggesting creating two sub-bit-streams for each slice and encrypt the headers sub-bit-stream. Assume all five macroblocks are non-intra.

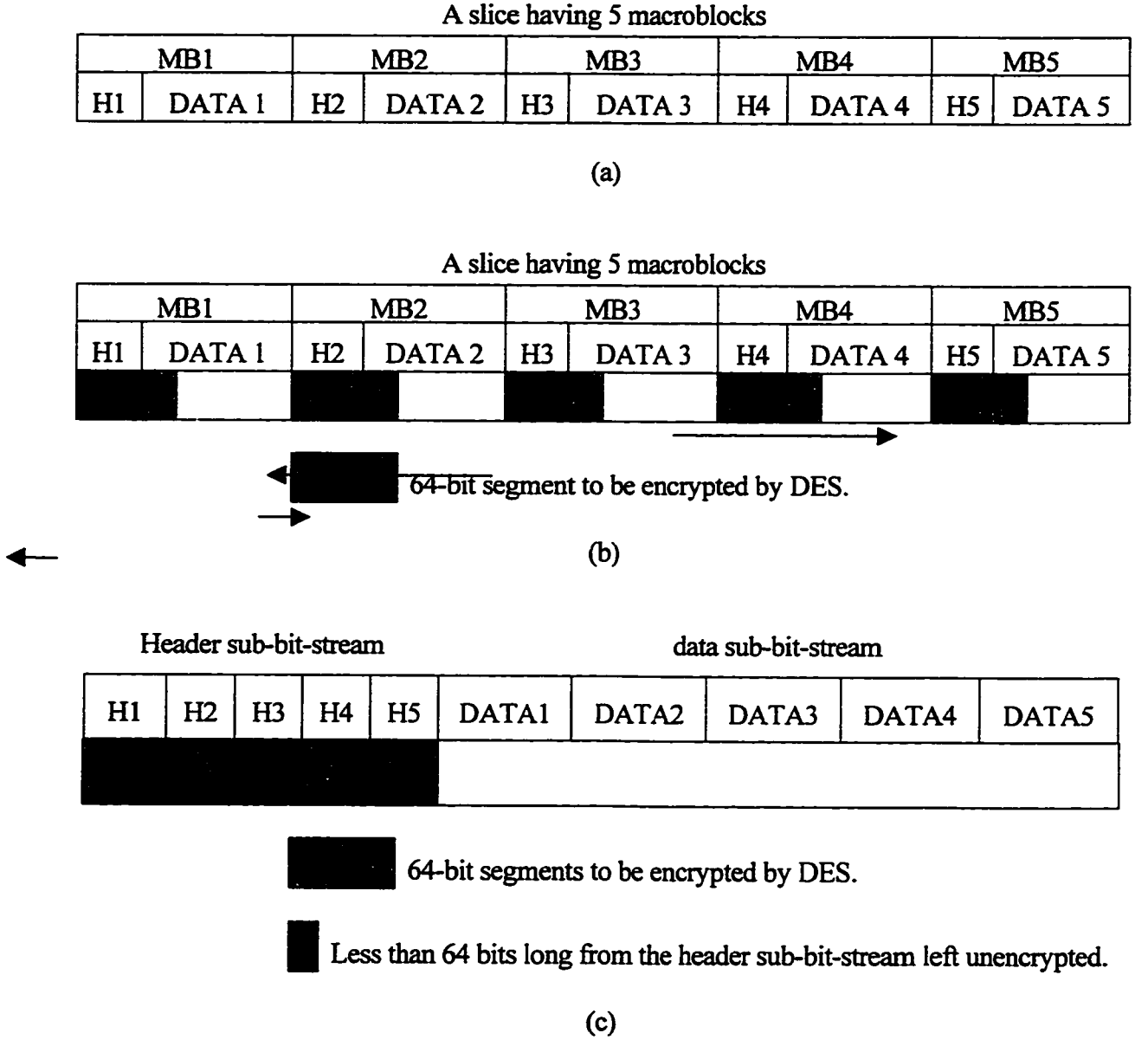
The first choice involves encrypting less data than the second choice. However, it produces a bit-stream that does not conform to the MPEG-I standard. To be able to decode this bit-stream the position of the data of all the MB's in each slice must be transmitted as an overhead information. In this case, decoding of each slice alternates between decoding the header sub-bit-stream and the data sub-bit-stream. It starts by decoding the header of the first MB from the header sub-bit-stream then switches to decode the data of the MB from the data sub-bit-stream. The same process is repeated for all MB's in a slice.

On the other hand, the bit-stream resulting from the second choice conforms to the MPEG-I standard; hence it does not require the transmission of any overhead information or restructuring at the decoder. Also, this method will introduce more disguising in the MPEG-I bit-stream, since the most upper-left DCT-coefficients of the first block in each P- or B-

macroblock will also be disguised. These coefficients are the DC and the low-order DCT coefficients, which carry most of the information in a block. For these reasons this choice was preferred over the first choice and it is implemented in this thesis. The inefficiency introduced by encrypting part of the non-intra macroblocks' data is calculated in section 3.5.2.

This encryption method is evaluated using the standard VMPEG 1.7d-Lite with encrypted bit-stream as well as using a standard MPEG-I decoder with zeroed bit-stream. Zeroing the non-intra MB's headers makes the standard decoder unable of decoding the resulting video. To overcome this difficulty, motion vector confusion and Huffman code re-synchronization have been used. Each MB_TYPE VLC code existing in the header indicates the kind of the motion vector used to predict the macroblock. For example, if the MB_TYPE VLC code in a B-macroblock header is (00010), then this macroblock is predicted by both forward motion and backward motion [17]. Motion confusion refers to replacing the MB_TYPE with a legal code other than the correct code. The new code is selected from Table (3.1). Table (3.1-a) is used with P-frames, while Table (3.1-b) is used with B-frames. This replacement of the MB_TYPE causes the decoder to treat the corresponding block improperly. The decoder may treat non-intra coded MB's as intra-coded MB's, or it may assign them the wrong motion type and get the wrong motion vectors for them. It may also use the wrong quantizer scale and assign them the wrong address. For example, in a B-frame, if the MB_TYPE VLC code is (00010), then, from Table (1.1-b), this macroblock is non-intra and both forward and

backward motion vectors are required to predict this macroblock. By replacing this code with (00011), the decoder will treat the macroblock as intra macroblock with no motion vectors.



MB1: The first macroblock, **H1**: The header of the first macroblock, **DATA1**: The data contents of the first macroblock.

Figure 3.8: An example illustrating two schemes to encrypt the headers of the non-intra macroblocks within a slice.

The above process will cause the decoder to loose bit-stream synchronization for a while. Depending on the bit-stream itself, bit-stream synchronization may be regained after a while. This causes a shift down in the DCT coefficients; i.e. high frequency coefficients are decoded and interpreted as low order coefficients. In order to simulate this effect, the first two low-order DCT coefficients are eliminated from the bit-stream. This is much less than the minimum encryption may cause. The encrypted 64-bit segment may contain more DCT coefficients. In order to keep the integrity of the bit-stream, the number of bits in the eliminated low frequency DCT-coefficients is counted and an equal number of zeros ("0" bits) is appended at the end of the corresponding slice.

Both of motion confusion and bit-stream re-synchronization have been implemented and integrated with the parser described before. This allows the standard MPEG-I decoder to get the most out of the unencrypted part of the bit-stream without attempting decrypting the bit-stream. It is expected that real encryption disguises the MPEG video bit-stream much more than this zeroing emulation does. However, real encryption of the MB's headers and using standard MPEG-I decoder may not get the most of the unencrypted data.

Encrypting all I-MB's and the headers of non-intra MB's is applied to all the video test clips. The VMPEG 1.7d-Lite decoder with the encrypted bit-stream produced similar results as the standard decoder with the modified zeroed bit-stream. For whole video test clips used in this thesis, encrypting P- and B-macroblocks' headers in addition to all I-macroblocks disguises the MPEG video completely even the motion in the video clip. Encrypting the *"Fibon"* video using this method disguises the motion and most of the information that the method of

encrypting I-macroblocks could not hide. In Figure (3.9), (a) is frame 79 of *"Fibon"* video clip, (b) is frame 23 *"Para"* video clip, (c) is frame 216 of *"Fibon"* video clip and (d) is frame 34 *"Para"* video clip. Figure (3.9-a and –c) show frames 79 and 216 of *"Fibon"* sequence after encryption using this method. The improved security level is clear when these two frames are compared to the corresponding frames shown in Figure (3.5-a and -c). Figure (3.9-b and -d) show frames 23 and 34 of the *"Para"* video clip after decoding the encrypted video by the VMPEG 1.7d-Lite decoder. When the video sequence was played, it was clear that further encrypting P- and B-macroblocks' headers disguises the motion of the paratroopers. The results of the other video clips are shown in the Appendix. These results have been verified by testing them by several colleagues.

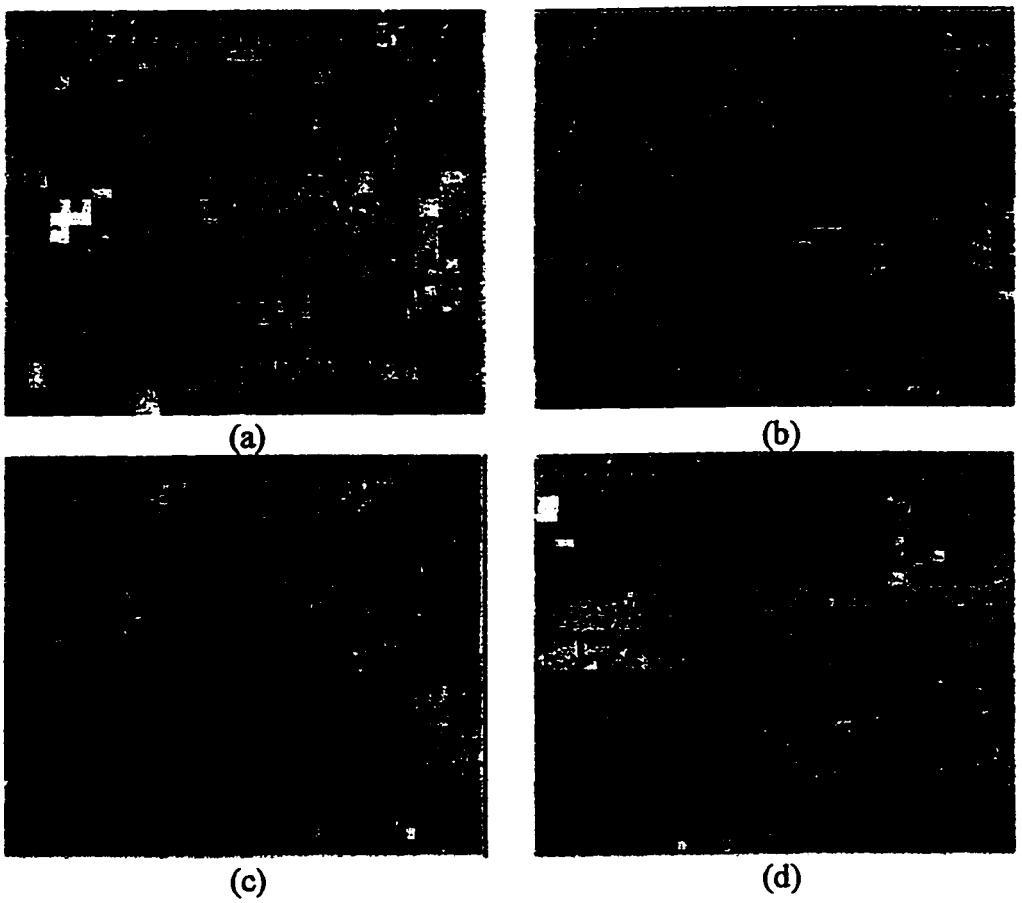


Figure 3.9: Encrypting P- and B-macroblocks' headers in addition to all I-MB's. These frames are obtained by decoding the encrypted video by the VMPEG 1.7d-Lite decoder.

3.5.2 Processing Time Calculations

The results shown in the previous section indicates that the method of encrypting the headers of P- and B-macroblocks in addition to all I-macroblocks achieves full disguising for naturally different MPEG-I video sequences. The cost is the increase in the processing time. Table (3.10) lists the fraction of video data being encrypted, τ , for each video clip encrypted by encrypting the headers of the P- and the B-macroblocks in addition to all I-macroblocks. From Table (3.10) the fraction of video data being encrypted, τ , achieved in this case varies

between 0.36 and 0.79. In the case of the *"Fibon"*, *"Para"* and *"Creamedgates"* video clips, the fraction of video data being encrypted, τ , is high. This is because these three video clips have high motion content that is predictable and evenly spread within each frame. For example the motion in the *"Fibon"* video clip is translational motion, which is highly predictable by motion compensation. This highly increases number of non-intra macroblocks in the video and decreases their contents (the prediction error). The *"Kangaroo"* video clip also has high motion contents, but this motion is limited to a small portion of each frame, which is the Kangaroo body only.

Video Clip Name	**Number of non-intra macroblocks**	**Total Number of bits in all I-MB's**	**τ by further encrypting all the headers of non-intra macroblocks**
"Fibon"	1904785	90667842	0.764
"Para"	23114	990381	0.786
"Creamedgates"	52384	1378766	0.771
"Gulf-War"	35608	3098121	0.419
"Kangaroo"	7750	1371729	0.403
"Fish"	13138	2071041	0.356
"Falcon"	7412	2041790	0.430
"OJ-Chase"	122442	5428411	0.583

Table 3.10: The fraction of video data being encrypted, τ , for the methods of encrypting all I-MB's and the method of further encrypting the headers of non-intra MB's.

To compare the saving in the processing time achieved by the three methods discussed in chapters 2 and 3 so far, Table (3.11) lists τ for these methods for all the video clips used in the thesis. τ in the method of encrypting I-frames only varies between 0.13 and 0.22. In the

method of encrypting all I-macroblocks τ varies between 0.24 and 0.35. Finally, by further encrypting the headers of P- and B- macroblocks τ varies between 0.36 and 0.79. So, τ increases as we move from the first method to the third one. On the other hand, the security level increases as we go from the first to the third method as demonstrated by the conducted experiments. The high jump in the saving in the processing time for the *"Fibon"* and the *"Para"* video clips between the second and the third methods is due to the nature of the motion in these sequences as explained before.

Video Clip Name	**τ by encrypting I-frames only**	**τ by encrypting I-MB's in All Frames**	**τ by further encrypting all the headers of non-intra macroblocks**	**The increase in τ caused by further encrypting the headers**
"Fibon"	0.220	0.326	0.764	0.438
"Para"	0.172	0.315	0.786	0.471
"Creamedgates"	0.195	0.306	0. 771	0.372
"Gulf-War"	0.191	0.242	0.419	0.177
"Kangaroo"	0.179	0.296	0.403	0.107
"Fish"	0.131	0.253	0.356	0.103
"Falcon"	0.198	0.349	0.430	0.081
"OJ-Chase"	0.186	0.239	0.583	0.344

Table 3.11: Comparison of τ between the three methods of selective encryption discussed so far.

From Table (3.11), τ for the method of encrypting all I-MB's and all headers of non-intra macroblocks is between 0.36 and 0.79. This is due to the high number of non-intra

macroblocks.' On the other hand, this method disguises the video completely. Section 3.6 modifies this method in order to reduce the processing time and hence reducing τ .

As discussed previously, the method of encrypting 64-bit segment from the header of each non-intra macroblock introduces inefficiency. This inefficiency is caused by encrypting part of the non-intra macroblock data in addition to the header where both the header and this part of the data make a 64-bit segment. Since the purpose is to encrypt the headers, encrypting part of the data introduces some inefficiency. This inefficiency is calculated and statistics have been conducted to calculate the data content of these headers. Table (3.12) calculates the inefficiency in the method of encrypting all I-MB's as well as the headers of the non-intra macroblocks. This inefficiency varies from one video to another. For the "*para*" video this inefficiency is 39.13. This means that 39.13 of the encrypted data are included inefficiently. Some of the videos such as "*para*" and "*OJ-Chase*" have high inefficiency compared to other clips. This indicates that the headers of the non-intra macroblocks contain fewer bits. This results because the header contains motion vectors (MV) for forward motion only or for backward motion only. However, this method of encryption is preferred upon the second one as discussed thoroughly in section 3.5.2.

Video Clip Name	Inefficiency (%)
"Fibon"	37.56
"Para"	39.13
"Creamedgates"	38.31
"Gulf-War"	28.62
"Kangaroo"	16.62
"Fish"	19.11
"Falcon"	12.07
"OJ-Chase"	36.47

Table 3.12: The inefficiency, in the method of encrypting all I-MB's as well as the headers of the non-intra macroblocks, introduced by encrypting part of the data of the non-intra macroblocks.

3.6 Alternative encryption/decryption

In order to reduce the processing time, other methods of encrypting an MPEG video sequence are introduced in this section. These methods rely on alternate encryption of the macroblocks. In the first method, the I-MB's in all frames are alternatively encrypted; i.e., the first I-MB is encrypted, the second is not encrypted, the third is encrypted, and so on. In the second method, the I-MB's are alternatively encrypted in addition to all the headers of the non-intra macroblocks. In the third method, I-MB's are alternatively encrypted and the headers of the non-intra macroblocks are alternatively encrypted too.

3.6.1 Encrypting I-MB's alternativley

In this method the I-MB's are encrypted alternatively. This method will not cause out of synchronization for the receiver. The transmitter will encrypt the first I-MB and leave the next one, then encrypt the third one and leave the fourth one, and so on. The encryptor will use a counter set at the beginning of each slice to keep track of the encrypted I-MBs. Similarly, the receiver will decrypt the first I-MB and decode the second one as it is, decrypt the third I-MB and decode the fourth one, and so on. Also, the decryptor will use a counter to keep track of the encrypted I-MB's. The only thing that both parties have to agree upon is to encrypt the odd I-MB's or the even ones. It is assumed in this thesis that the odd I-MB's are encrypted. This implies that half the I-MB's in the MPEG video sequence and nearly half the content of I-MB's are encrypted. This method has been applied on all the test video clips used in the thesis. The resulting encrypted MPEG video sequences were decoded using the VMPEG 1.7d-Lite decoder. The results showed no noticeable difference between this method and the method of encrypting all I-MB's in all frames from the security level viewpoint. For example in the *"Para"* video sequence, the motion of the paratrooper is identified, as was the case when encrypting all I-MB's. These results are shown in the Appendix.

To compute the processing time of this method, the number of encrypted I-MB's and the data contained by these I-MB's are calculated for each video sequence used in the thesis. Table (3.13) lists these statistics in addition to the fraction of the video data being encrypted, τ , for this method.

Video Clip Name	Number of encrypted odd I-MB's	Number of bits in these macroblocks	τ by encrypting I-MB's alternatively	τ by encrypting all I-MB's
"Fibon"	154452	45277776	0.163	0.326
"Para"	2699	495631	0.158	0.315
"Creamedgates"	4072	688379	0.153	0.306
"Gulf-War"	4214	1553628	0.121	0.242
"Kangaroo"	2161	688226	0.149	0.296
"Fish"	3371	1034583	0.126	0.253
"Falcon"	2941	1025710	0.175	0.349
"OJ-Chase"	11844	2713584	0.119	0.239

Table 3.13: Statistics showing the τ for both methods of encrypting all I-MB's and half of them.

The results in Table (3.13) show how the method of encrypting half I-MB's reduces the processing time by half while the security level is relatively the same as have been verified by some colleagues. The next step is to encrypt all the headers of the non-intra macroblocks and half the I-MB's.

3.6.2 Encrypting I-MB's alternativley and all the headers of the non-intra macroblocks

In section 3.5.1, we concluded that encrypting half the I-MB's has no noticeable effect on the security level compared to encrypting all the I-MB's while the processing time is reduced by nearly half. Furthermore, in section 3.5, encrypting all I-MB's and all the headers of the non-intra macroblocks disguised the MPEG video sequences completely. Hence, by combining

these schemes together, a new scheme of encrypting all the headers of non-intra macroblocks and half the I-MB's is implemented. This scheme is expected to disguise the MPEG video sequences completely with reducing the processing time compared to the method of section 3.5. A counter is used to keep track of the encrypted I-MB's at both the encryptor and the decryptor.

This method has been applied on all the video sequences used in the thesis. The encrypted MPEG video sequences have been decoded using VMPEG 1.7d-lite decoder and the results show that this method disguises the video sequences completely. One can not tell any noticeable difference between the sequences encrypted using this method and the ones resulting from the method of section 3.5. These results are shown in the Appendix. These results have been evaluated by several colleagues and they got no information from the encrypted video clips.

The processing time of this method is calculated by computing the number of bits contained by half the I-MB's and the 64-bit segments of the headers of the non-intra macroblocks. The fraction of the video data being encrypted, τ , for this method is shown in Table (3.14) together with the fraction of the video data being encrypted, τ , obtained by the method of section 3.5 (i.e. encrypting all I-MB's as well as the headers of all non-I-MB's).

Video Clip Name	τ by encrypting all the headers of non-intra macroblocks and all the I-MB's	τ by encrypting all the headers of non-intra macroblocks and half the I-MB's	The decrease in τ
"Fibon"	0.764	0.455	0.309
"Para"	0.786	0.628	0.158
"Creamedgates"	0. 771	0.618	0.153
"Gulf-War"	0.419	0.299	0.120
"Kangaroo"	0.403	0.256	0.147
"Fish"	0.356	0.229	0.127
"Falcon"	0.430	0.256	0.174
"OJ-Chase"	0.583	0.464	0.119

Table 3.14: The fraction of the video data being encrypted, τ , for both the method discussed in section 3.5 and the method of encrypting half I-MB's and all the headers of the non-intra macroblocks.

Table (3.14) shows that the method of encrypting both half the I-MB's and all the headers of the non-intra macroblocks has a τ between 0.23 and 0.63. The decrease in the fraction of the video data being encrypted, τ , compared to the method of section 3.5 is between 0.12 and 0.30. For example, if the thorough encryption of an MPEG video sequence takes one hour, this method disguises the sequence completely in a time between 13.8 and 37.8 minutes. Hence, the gain is between 46 and 22 minutes.

Table (3.15) calculates the inefficiency in this method that is introduced by encrypting part of the data in the non-intra macroblocks. The inefficiency here is expected to be higher than in the method of section 3.4. This is due to the decrease in the total number of bits being encrypted because of the reduction in the number of encrypted I-MB's while the size of the

data that causes the inefficiency remains the same. However, as stated previously, even though these bits introduce inefficiency they add to the security level.

Video Clip Name	Inefficiency (%)
"Fibon"	47.76
"Para"	44.59
"Creamedgates"	49.85
"Gulf-War"	40.15
"Kangaroo"	26.21
"Fish"	29.67
"Falcon"	20.25
"OJ-Chase"	45.86

Table 3.15: The inefficiency, in the method of encrypting half the I-MB's as well as the headers of all the non-intra macroblocks, introduced by encrypting part of the data of the non-intra macroblocks.

3.6.3 Encrypting alternatively both the I-MB's and the headers of the non-intra macroblocks

In this method, encryption is applied on half the I-MB's and to the headers of half the non-intra macroblocks. For those video sequences where the number of non-intra macroblocks is large, this method reduces the processing time considerably. In this method, two counters are used, one to keep track of the encrypted I-MB's while the second one is to keep track of the encrypted headers of the non-intra macroblocks. This method has been applied on all the video clips used in this thesis. The encrypted video sequences have been decoded using the VMPEG 1.7d-lite decoder. The decoded video clips show totally disguised MPEG video

sequences. These results are shown in the Appendix. These results have been tested by several colleagues and they reveal no information from the encrypted video clips.

Video Clip Name	τ by encrypting all the headers of non-intra macroblocks and all the I-MB's	τ by encrypting all the headers of non-intra macroblocks and half the I-MB's	τ by encrypting the headers of half the non-intra macroblocks and half the I-MB's
"Fibon"	0.764	0.455	0.382
"Para"	0.786	0.628	0.393
"Creamedgates"	0. 771	0.618	0.378
"Gulf-War"	0.419	0.299	0.210
"Kangaroo"	0.403	0.256	0.202
"Fish"	0.356	0.229	0.178
"Falcon"	0.430	0.256	0.216
"OJ-Chase"	0.583	0.464	0.292

Table 3.16: The fraction of the video data being encrypted, τ , for three methods.

Table (3.16) shows the fraction of the video data being encrypted, τ , for this method and compares it with the method of section 3.5 and the method discussed in section 3.6.2 where half the I-MB's and the headers of all the non-intra macroblocks are encrypted. The fraction of the video data being encrypted, τ , for this method is less than 0.40. This implies that the gain is, at least, 60%. For example, if the thorough encryption of an MPEG video sequence requires one hour, then this method requires a maximum time of 24 minutes. Furthermore, for some sequences such as the *"Fish"* video sequence where the number of non-intra macroblocks is small, this method requires 11 minutes only.

Table (3.17) calculates the inefficiency in the proposed method introduced by including part of the non-intra macroblocks' data in the encrypted part of the header. This inefficiency is expected to be very close to the one in section 3.5.2. This is because both the encrypted data part of the non-intra macroblocks and the total encrypted data is reduced by, nearly, the same factor.

Video Clip Name	Inefficiency (%)
"Fibon"	37.58
"Para"	39.24
"Creamedgates"	42.62
"Gulf-War"	28.61
"Kangaroo"	16.59
"Fish"	19.04
"Falcon"	12.02
"OJ-Chase"	36.46

Table 3.17: The inefficiency, in the method of encrypting half the I-MB's as well as the headers of half the non-intra macroblocks, introduced by encrypting part of the data of the non-intra macroblocks.

For all the test video clips used in this thesis, this method of encrypting half the I-MB's and the headers of half the non-intra macroblocks disguises the video clips completely. However, in the "*Fibon*" video clip, there is a series of numbers (1,2,3,4,5, ... etc.) appear in the video clip. In the decoded-encrypted "*Fibon*" video clip using the above method, the numbers (2,3 and 4) appear while all other numbers in the series are still disguised. This does not affect the security level because in the 6773 frames of the "*Fibon*" video clip, a series of numbers

appear in several parts of the video clip. Only at one position, the numbers 2,3 and 4 appear in the encrypted "*Fibon*" video clip. These numbers are in motion in the original video clip. However, if the MPEG video clip is for a critical financial meeting of an organization, the numbers in the video clip will not be in motion and hence they will be completely disguised.

A further experiment is conducted to reduce the fraction of the video data being encrypted, τ . This experiment is to encrypt half the I-MB's and the headers of third the non-intra macroblocks. This experiment has been conducted and the VMPEG 1.7d-lite decoder has been used to decode the encrypted video clips. The results show a clear degradation in the security level. The motion in the encrypted/decoded video clips is very clear. These results are shown in the Appendix.

3.7 Encrypting 64-bit segment from every other macroblock

3.7.1 Experimental methodology and results

The results from the previous sections show that encrypting I-MB's in addition to the headers of the non-intra macroblocks of MPEG-I video bit-stream ensures high security level for the encrypted video. Also, alternative encryption reduces the ratio of video data being encrypted, τ , without affecting the security level. All these observations lead to a new scheme to selectively encrypt the MPEG video bit-stream with the minimum processing time. This scheme is to encrypt the first 64-bit segment from every other macroblock regardless of its type.

The same parser used before is used here to parse and pass the first 64-bit segment from every other macroblock to the DES algorithm to be encrypted. A counter is used to keep track of the encrypted macroblocks. This counter is set at the beginning of each slice slice. Also, similar counter is used by the receiver to keep track of the encrypted macroblocks.

Encrypting the headers of the non-intra macroblocks hides the motion information. However, encrypting 64 bits from an I-MB hides the whole macroblock even though the data of the I-MB is not encrypted. To understand this, the coding process of the DC component of the DCT block has to be explained first. The DC value is coded differentially according to equation (3.3) [10, 42].

$$\Delta DC = DC - P \tag{3.3}$$

Where DC is the DC value of the block being encoded and P is the DC value of the neighboring block just encoded and it is used as a prediction of the DC component of the block being encoded. This technique is called predictive DPCM [42]. The receiver receives ΔDC and it uses the DC value of the neighboring block that has just been decoded as P. Then the receiver adds these two quantities to get the DC component of the block being decoded. This process is done for the luminance (Y) and the chrominance (Cr and Cb) blocks separately. This is illustrated in Figure (3.10) where **Y1** is the first luminance block, **Y2** is the second luminance block...etc. and **Cb** and **Cr**: are the chrominace (color) blocks. This coding process for the DC coefficient is performed between blocks of the same type (Y, Cr, or Cb) in the same I-MB and in consecutive I-MB's.

Hence, the encrypted 64-bit segment from the I-MB contains at least the DC coefficient of the first luminance block. So, encrypting this segment will change the value of the prediction for all the following blocks. So, referring toFigure (3.10), the new scheme encrypts both I-MB(i-1) and I-MB(i+1) macroblocks and leaves I-MB(i) macroblock unencrypted. Even though I-MB(i) is unencrypted but it will not reveal any useful information because the prediction coming from I-MB(i-1) is not the right one because it is encrypted. Losing the low-frequency components of the DCT block causes loosing the whole block.

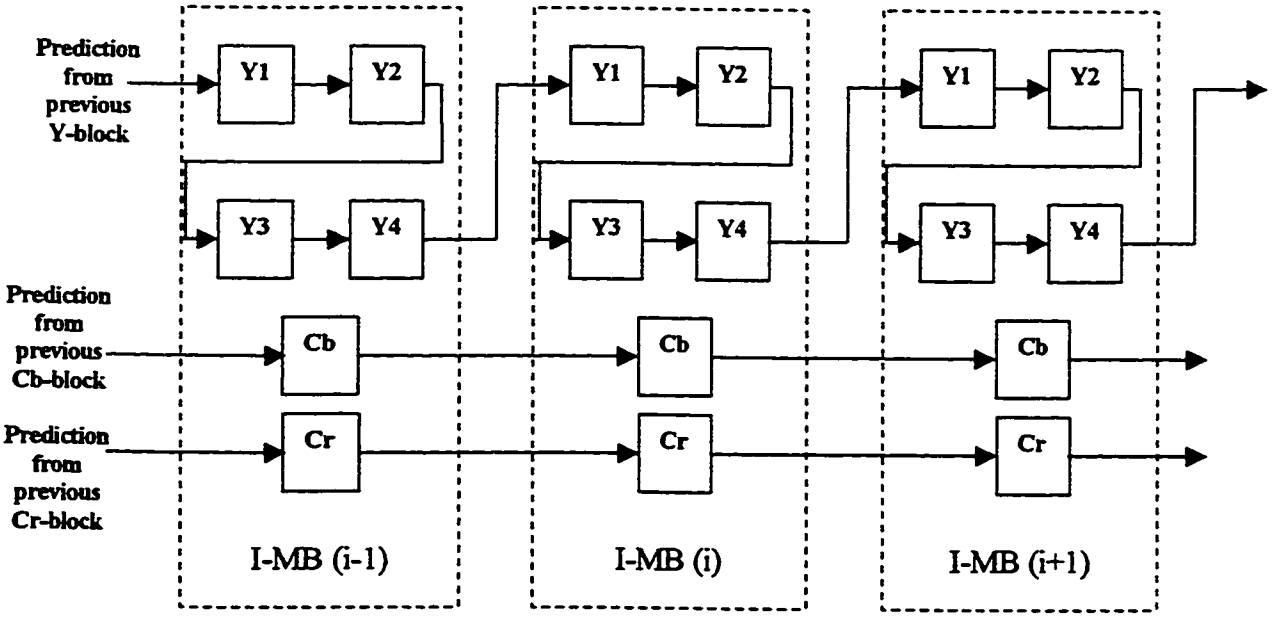


Figure 3.10: The process of differentially encodeing the DC value in the blocks of I-MB's. (Similar figure exists in [42] as Figure 5.5).

The method of encrypting the first 64 bits from every other macroblock has been applied on the test video clips used in this thesis. The encrypted video clips using this scheme have been decoded using the VMPEG 1.7d-lite decoder. The decoded sequences show a completely

disguised video clips. These results are shown in the Appendix. Testing these results by several colleagues gives the conclusion that this method disguises the video completely.

This scheme has a major advantage over the previous discussed schemes. In this scheme, the receiver does not have to apply the decision process discussed in Section 3.3.3. Hence, it does not introduce any decoding delay at the receiver and the bit-stream is compatible with MPEG-I standard.

3.7.2 Processing Time Calculations

The scheme of encrypting 64-bit segment from each macroblock ensures fully secure transmission of MPEG video sequences. To calculate the fraction of the video data being encrypted, τ, half the total number of macroblocks in each video clip has to be calculated and then multiplied by 64 and then compared to the data in all frames. The number of macroblocks in each test video clip used in this thesis is listed in Table (3.3). So, these calculations give the results in Table (3.18). From the table the fraction of the video data being encrypted, τ, varies between 0.0726 and 0.290. Hence, this scheme achieves a considerable reduction in the required processing time compared to thorough encryption. For example, if thorough encryption of a certain MPEG video sequence requires one hour, hen this scheme ensures the same security level with processing time between 4 and 18 minutes only.

Hence, fully disguised MPEG video sequences that can be obtained by selective encryption with the minimum τ is obtained by the method of encrypting 64-bit segment from every other macroblock in the video clip.

Video Clip Name	The fraction of the video data being encrypted, τ
"Fibon"	0.255
"Para"	0.290
"Creamedgates"	0.276
"Gulf-War"	0.110
"Kangaroo"	0.083
"Fish"	0.0778
"Falcon"	0.0726
"OJ-Chase"	0.206

Table 3.18: The fraction of the video data being encrypted, τ , for the method of encrypting 64-bit segment from every other macroblock.

3.8 Summary

In this chapter, two classifications of the MPEG video sequence have been proposed. The first classification is to consider all I-MB's in all frames (I, P and B) as the independent part of the MPEG video bit-stream. This method of encrypting all I-MB's in all frames has been tested and compared to the method of encrypting I-frames only. The experimental results show an improvement in the security level. However, the motion is still clearly recognized and the processing time increases. This increase in the processing time or, equivalently, this

degradation in the processing time depends on how many macroblocks in P- and B-frames are encoded as intra and on how many bits these macroblocks contain. As concluded previously, if the bits contained by I-MB's outside the intra frames are less than 50% of the total bits in all intra macroblocks, then the degradation in the processing time is small. This method has been improved further by encrypting half the I-MB's. The security level remains almost the same, while the processing time has been reduced by, nearly, half.

The content of the MPEG-I video clips encrypted by this method is disguised except for the motion, where one can recognize the motion of a body in the encrypted video clip. This motion information exists in the headers of the non-intra macroblocks that exist in the non-intra frames (P and B). So, disguising this information will remove the motion information from the bit-stream and hence improves the security level. This led to the second classification of the MPEG video sequence. This classification is to consider, in addition to the I-MB's, the headers of the non-intra macroblocks as the independent part of MPEG-I video bit-stream. This is because of the existence of the motion information in this part of the MPEG video sequence. So, the motion information has been disguised by further encrypting the headers of P- and B-macroblocks in addition to all I-macroblocks in all frames. The results of this method show a completely disguised transmission of MPEG video sequences. However, the processing time increases compared to encrypting I-macroblocks only. The fraction of the video data being encrypted, , achieved by encrypting I-macroblocks only is between 0.23 and 0.35 while by further encrypting the headers of P- and B- macroblocks the fraction of the video data being encrypted is between 0.36 and 0.79. This method has been

improved by encrypting half the I-MB's and all the headers of the non-intra macroblocks. The security level remains the same while the fraction of the video data being encrypted, τ, decreases to be between 0.23 and 0.63. By encrypting half the I-MB's and the headers of half the non-intra macroblocks, the fraction of the video data being encrypted, τ, decreases to be between 0.18 and 0.39. However, the security level remains the same where the MPEG video sequence is disguised completely. Further reduction in the processing time can be achieved by encrypting half the I-MB's and the headers of third the non-intra macroblocks. However, decoding the encrypted video clips using this method show a considerable degradation in the security level.

The last suggested scheme is to encrypt 64-bit segment from every other macroblock in the MPEG video clip. The fraction of the video data being encrypted, τ, achieved by this scheme is between 0.0726 and 0.290. As a result, we conclude that the minimum processing time, with keeping the desired security level, is obtained by encrypting 64-bit segment from every other macroblock.

These methods discussed in this chapter achieve a completely secure transmission of MPEG digital video sequences. This is valid as far as the eavesdropper does not perform cryptanalysis on the encrypted part of the bit-stream. In this case, a key management protocol has to be designed carefully so that the crpytanalyst will not have enough time to do cryptanalysis on the MPEG bit-stream.

Chapter 4

Analyzing the Security Level of the Selective Encryption Methods

4.1 Introduction

In the previous chapter, several methods for selective encryption of MPEG-I bit streams were suggested and several experiments were conducted in order to evaluate the security level achieved by each of these methods. Throughout these experiments, no attempts were made to decrypt the encrypted part of the MPEG-I bit stream, and all effort was directed toward

simulating the effect of decoding the video sequence from the available unencrypted parts of the bit stream. For this purpose, a priori knowledge of which part of the bit stream is encrypted and which part is not was assumed. Also, these experiments reveal that the method of encrypting all I-MB's, the method of encrypting all the headers of the non-intra MB's in addition to encrypting all the I-MB's and the method of encrypting the header of every other macroblock are more resilient to such attack than the other methods, which makes them good target of more serious attacks. Usually, these serious attacks are very costly and time consuming, since they involve a lot of computations and trials. They are launched only if the video material contained within the encrypted bit stream is of very sensitive nature. Since in this thesis we are only concerned with very sensitive real-time applications, it is expected that the cryptanalyst will be willing to pay as much as he can and will use all the computation power available to him in order to decrypt the video bit stream. An example of such applications is military, where a leader is transmitting an MPEG video sequence that explains a classified training procedure or some drills of attack on enemy basis. In this case, the enemy is willing to pay whatever it costs in order to view the original MPEG video sequence.

Simple attack through decoding the unencrypted part of the bit stream will not provide satisfactory results. So, upon decoding the first very few frames of the encrypted sequence, the eavesdropper will decide either not to continue decoding the bit stream or to launch a more serious attack against the encrypted part of the bit stream. For the method of encrypting all I-macroblocks in all frames, the only attacking method involves the use of cryptanalysis techniques. However, for the method of further encrypting non-intra macroblocks' headers,

attacks involve guessing the contents of the non-intra frame headers in addition to the use of cryptanalysis techniques. The contents of these headers are specified in the MPEG-I standard [17]; hence, they can be easily guessed. Assuming that the secured MPEG-I bit stream is available, the goal of the cryptanalysis is to obtain the encryption key, which was used in encrypting the bit stream. The harder to get the encryption key the harder is the cryptanalysis.

Section 4.2 of this chapter discusses the time required for an eavesdropper to correctly guess these headers and the time required for him to resynchronize the bit stream. Section 4.3 discusses how the DES key is going to be transmitted within the MPEG video bit stream. Section 4.4 presents an estimation of the time and cost of attacking the encryption/decryption keys. Section 4.6 is a summary.

4.2 Attacking the Proposed Schemes without Crypt-analysis

In Chapter 3, three schemes have been proposed and tested. The first scheme is encrypting I-macroblocks in all frames. This scheme enhances the security level compared to encrypting I-frames only. However, the eavesdropper can extract motion information from the video by designing a special decoder that skips over the encrypted parts of the bit stream. On the other hand, further encrypting P- and B-macroblocks' headers hides this information even if the eavesdropper designed his decoder to skip over the encrypted part of the bit stream as shown in Chapter 3. Reduction in the video data being encrypted is achieved by applying alternative encryption. The third method shows that encrypting the first 64-bit segment from every other

macroblock provides fully disguised MPEG video sequences. So, we concluded that the last two schemes disguise the transmitted video completely against unauthorized eavesdroppers.

4.2.1 Guessing the Bit stream Header

The parameters in the macroblock headers have certain values that are defined in the MPEG-I standard [17]. These parameters are also listed in Table (4.2). If the eavesdropper guesses the value of these parameters correctly, then he will be able to identify the motion in the decoded video based on the non-encrypted part of the bit stream. Hence, via guessing the eavesdropper can remove the effect of encrypting the non-intra macroblocks' headers. In this section, the time required for the eavesdropper to correctly guess the headers is estimated.

The decoding syntax of the macroblock header is shown in Table (4.1) [17]. In this table, the macroblock parameters appear in "bold face". Each of these parameters has well known possible values and its maximum length in bits is also known. Table (4.2) lists each parameter along with its possible length and number of possible values it may assume [17]. As it was stated in Chapter 3 of this thesis, macroblock header encryption starts after the **Macroblock_type** parameter. This allows the decoder to distinguish between intra and non-intra macroblocks. Hence, the first five parameters of Table (4.2) are known to the cryptanalysis. The value of the **Macroblock_type** parameter depends on the picture (I, P or B). The **Macroblock_type** indicates whether the parameters listed in Table (3.1) are encoded or not [17]. If the **Macroblock_type** indicates that the value of one of these parameters is encoded in the bit stream, then this value must be decoded from the bit stream and used in the

decoding process. These parameters are *Macroblock_quant*, *Macroblock_motion_forward*, *Macroblock_motion_backward*, *Macroblock_pattern*, and *Macroblock_intra*. The *Macroblock_quant* indicates that a new quantizer level is encoded in the bit stream. If *Macroblock_motion_forward* is 1 and *Macroblock_motion_backward* is 0 this indicates that the macroblock is forward predicted and its motion vectors are encoded in the bit stream. Similarly, if *Macroblock_motion_forward* is 0 and *Macroblock_motion_backward* is 1 this indicates that the macroblock is backward predicted and its motion vectors are encoded in the bit stream. If both *Macroblock_motion_forward* and *Macroblock_motion_backward* are 1 this indicates that the macroblock is bi-directionally predicted and its forward and backward motion vectors are encoded in the bit stream. The *Macroblock_pattern* indicates which of the six blocks comprises the macroblock are encoded in the bit stream. *Macroblock_intra* indicates whether the macroblock is intra or predicted.

Let us assume that the eavesdropper knows that all I-macroblocks in all frames are encrypted as well as the non-intra macroblocks' headers or he knows that 64-bit segment from every other macroblock is being encrypted. To find the time required for the eavesdropper to guess correctly the headers of the macroblocks in P- and B-pictures, we must find the number of all combinations of parameters in the header. The number of these combinations in a P-picture is different than that in a B-picture.

```
macroblock ()     {
        while ( nextbits() == '0000 0001 111' )
                macroblock_stufing
        while ( nextbits() == '0000 0001 000' )
                macroblock_escape
        macroblock_address_increment
        macroblock_type
        if ( macroblock_quant )
                quantizer_scale
        if ( macroblock_motion_forward )    {
                motion_horizontal_forward_code
                if ( (forward_f != 1 ) AND (motion_horizontal_forward_code != 0) )
                        motion_horizontal_forward_r
                motion_vertical_forward_code
                if ( (forward_f != 1 ) AND (motion_vertical_forward_code != 0) )
                        motion_vertical_forward_r
        }
        if ( macroblock_motion_backward )  {
                motion_horizontal_backward _code
                if ( (backward _f != 1 ) AND (motion_horizontal_backward_code != 0) )
                        motion_horizontal_backward_r
                motion_vertical_backward_code
                if ( (backward_f != 1 ) AND (motion_vertical_backward_code != 0) )
                        motion_vertical_backward_r
        }
        if ( macroblock-pattern )
                coded_block_pattern
        for (i=0 ; i<6 ; i++)
                block ( i )
        if ( picture_coding_type == 4 )
                end_of_macroblock
}
```

Table 4.1: The syntax of the macroblock header in MPEG-I bit stream.

The parameter		Length in bits	No. of Possible Values
Macroblock_stuffing		11	1
Macroblock_escape		11	1
Macroblock_address_increment		1-11	33
Macroblock_type		1-6	*
Quantizer_scale		5	31
Forward MV_x	*Motion_horizontal_forward_code*	1-11	33
	Motion_horizontal_forward_r	1-6	64
Forward MV_y	*Motion_vertical_forward_code*	1-11	33
	Motion_vertical_forward_r	1-6	64
Backward MV_x	*Motion_horizontal_backward_code*	1-11	33
	Motion_horizontal_backward_r	1-6	64
Backward MV_y	*Motion_vertical_backward_code*	1-11	33
	Motion_vertical_backward_r	1-6	64
Coded_block_pattern		3-9	63

Table 4.2: The length (in bits) and the number of possible values for each parameter in the macroblock header. (* The number of possible values for the **macroblock_type** parameter depends on the picture type.)

In a P-picture there is no backward motion. This motion is indicated by the field of **macroblock_motion_backward** in Table (4.1) for P-pictures where it is always '0'. So, the parameters **motion_horizontal_backward _code, motion_horizontal_backward_r, motion_vertical_backward_code** and **motion_vertical_backward_r** do not exist in the macroblock's header in a P-picture. Furthermore, the two values of the **macroblock_type** parameter '00011' and '000001' will not be guessed because they indicate that the macroblock is of type intra.

For each value of the **macroblock_type** parameter we will have different values of the other parameters in the header. Some parameters might have a value that affects the existence of another parameter such as the **motion_vertical_forward_code**, which affects the existence of the **motion_vertical_forward_r** parameter. If the former has a '0' value then the latter does not exist. Also, the later parameter is affected by a parameter from the picture header.

This outside-parameter is the **forward_f** that appears in the IF statements in Table (4.1). The eavesdropper knows the **macroblock_type** since it is not encrypted. Hence, he has to guess **Quantizer_scale, Forward MV_x, Forward MV_y** and **Coded_block_pattern** in case the **macroblock_type** indicates the existence of these parameters. As a clarification example, assume that the **macroblock_type** is (00010). This indicates the existence of all the four parameters as shown in Table (4.1) [17]. The **Quantizer_scale** can have 31 values while the **Coded_block_pattern** can have 63 values according to Table (4.2). Each of the **Forward MV_x** and the **Forward MV_y** parameters can have a value between –16 and +16 with two possible resolutions (1 or 0.5), depending on the **forward_f** parameter. The discussion illustrating these possible values is beyond the scope here and the reader is referred to [17] for more details of how to get these values. What we are concerned about here is the possible number of combinations.

Small motion vectors are more probable than large ones. Hence, the eavesdropper might guess the small ones first. To do that he will guess first '0' motion vector then '+1', then '-1' and so on up to '16'. To account for this pattern, a certain probability for each motion vector has to be assigned. This set of probabilities has been calculated for the '*Fibon*' video clip, which contains 6773 frames. The histograms for **Forward MV_x , Forward MV_y, Backward MV_x** and **Backward MV_y** are shown in Figure (4.1). It is clear from the figure that the small vectors are much more probable than large ones. Hence, the eavesdropper will have such model and then he will start his guessing process. For the **Forward MV_x** parameter, he will first guess '0' vector with resolution 1. If the decoded

macroblock is not right he will guess the '0' vector with resolution 0.5. If the decoded macroblock is still not the right one, he will guess the vector to be '+1' with resolution 1. He will continue this process until he gets the correct motion vector. However, the resolution whether it is half pel (0.5) or full pel (1) can be known from the header of the picture.

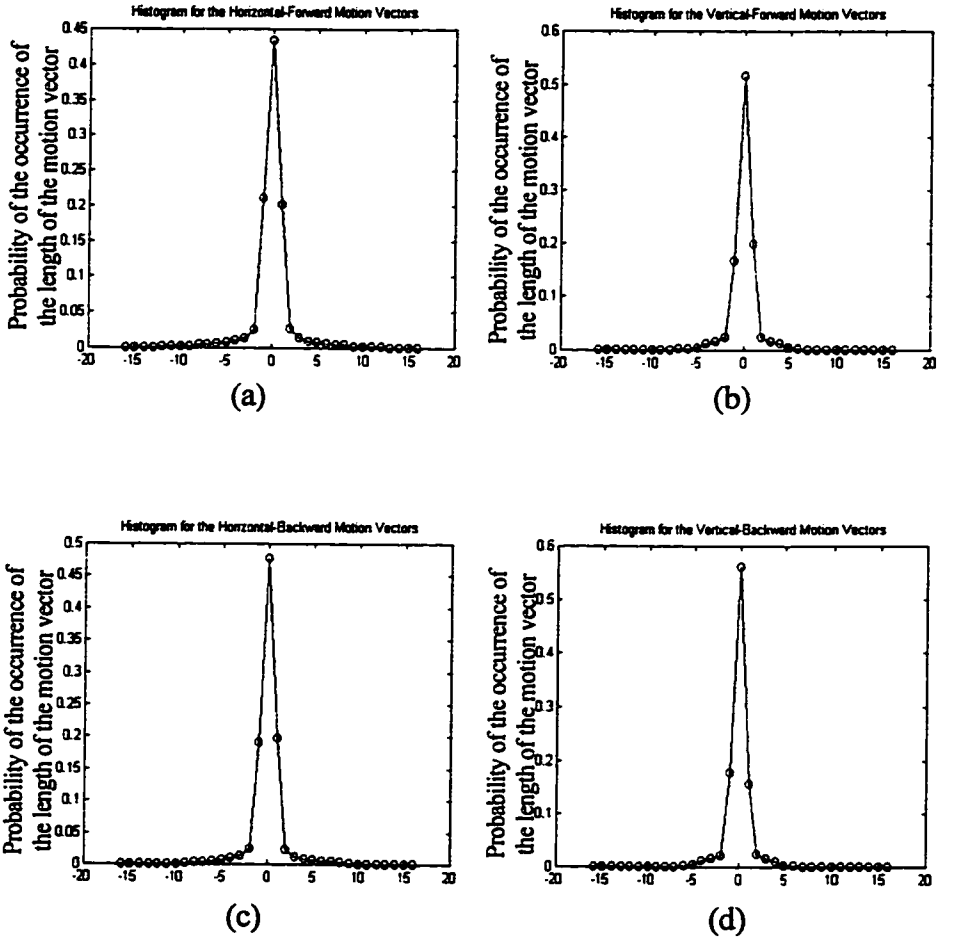


Figure 4.1: Histograms for the '*Fibon*' Motion Vectors (MV). (a) **Forward MV_x.** (b) **Forward MV_y.** (c) **Backward MV_x** and (d) **Backward MV_y.** (The X-axis represents the length of the motion vector.)

Let us assume that the two resolutions 1 and 0.5 are equiprobable. Also, assume that the sequence of motion vectors, the eavesdropper will follow are as follows: 0(1), +1(1), -1(1),

2(1),....., 16(1), -16(1). The parameter -1(1) means that the vector is -1 with resolution 1. This is if the picture header indicates 1-resolution. Let us first find the average number of times the eavesdropper has to decode the macroblock to get the correct **MV**. This average is given by equation (4.1).

$$\text{Average number of times to decode the macroblock} = \sum_{i=1}^{33} i \times P_i \qquad (4.1)$$

where:

P_1 = *Probability of decoding one time = Probability (motion vector is 0(1))*

P_2 = *Probability of decoding two times = Probability (motion vector is 1(1))*

.....

.....

P_{32} = *Probability of decoding 32 time = Probability (motion vector is 16(1))*

P_{33} = *Probability of decoding 33 time = Probability (motion vector is -16(1))*

So, if the motion vector is 0(1), then the eavesdropper will decode the macroblock one time and this occurs with probability equal to the probability of 0(1). Similarly, if the motion vector is 2(1), then the eavesdropper will decode the macroblock seven times and this occurs with probability equal to the probability of 2(1). P_i in equation (4.1) depends on the type of the motion vector, whether **Forward MV_x , Forward MV_y, Backward MV_x** or **Backward MV_y**. This probability depends on the model the eavesdropper will follow. We will assume Figure (4.1) as the model to have an estimate of the average time the

eavesdropper will guess the header correctly. According to this model, the average number of times, which the eavesdropper will decode the macroblock for each type of the motion vectors assuming that the macroblock will contain only one type at a time, is given by the following equations according to Figure (4.1).

$$\textit{Average number of times to decode one macroblock that contains Forward MV\_x only} = \sum_{i=1}^{33} i \times P_i = 2.9058 \textit{ times} \quad (4.2)$$

$$\textit{Average number of times to decode one macroblock that contains Forward MV\_y only} = \sum_{i=1}^{33} i \times P_i = 3.2293 \textit{ times} \quad (4.3)$$

$$\textit{Average number of times to decode one macroblock that contains Backward MV\_x only} = \sum_{i=1}^{33} i \times P_i = 2.6013 \textit{ times} \quad (4.4)$$

$$\textit{Average number of times to decode one macroblock that contains Backward MV\_y only} = \sum_{i=1}^{33} i \times P_i = 2.1028 \textit{ times} \quad (4.5)$$

Since, the **macroblock_type** parameter is not encrypted, then the eavesdropper will know which of the parameters: **Coded_block_pattern, Quantizer_scale, Backward MV** and **Forward MV** exist. Let us first find the average time required to guess the headers of all non-intra macroblocks in a P-picture correctly. According to Table (4.1), five code words for the **macroblock_type** parameter in a P-picture require the eavesdropper to guess the headers

since they indicate that the macroblock is non-intra. Table (4.3) gives the average number of times the eavesdropper has to decode the macroblock with a certain **macroblock_type** parameter. By assuming that these code words are equiprobable, the average number of times the eavesdropper has to decode a non-intra macroblock in a P-picture is given by summing the values for the P-picture in the sixth column of Table (4.3) and divides it by five to give *1080* times.

	macroblock_type	No. of possible values for Quantizer_scale	Average No. of possible values for Forward MV	Average No. of possible values for Backward MV	No. of possible values for Coded_block_pattern	Average No. of times to decode a macroblock with this **macroblock_type** parameter
P-picture	**1**	NO	2.9058×3.2293	NO	63	(2.9058×3.2293) ×(63/2)
	01	NO	NONE	NO	63	63/2
	001	NO	2.9058×3.2293	NO	NO	(2.9058×3.2293)
	00010	31	2.9058×3.2293	NO	63	(31/2) × ((2.9058×3.2293) ×(63/2)
	00001	31	NO	NO	63	(31/2) ×(63/2)
B-picture	**10**	NO	2.9058×3.2293	2.6013×2.1028	NO	(2.9058×3.2293) ×2.6013×2.1028
	11	NO	2.9058×3.2293	2.6013×2.1028	63	(2.9058×3.2293) ×2.6013×2.1028×(63/2)
	010	NO	NO	2.6013×2.1028	NO	2.6013×2.1028
	011	NO	NO	2.6013×2.1028	63	2.6013×2.1028×(63/2)
	0010	NO	2.9058×3.2293	NO	NO	(2.9058×3.2293)
	0011	NO	2.9058×3.2293	NO	63	(2.9058×3.2293) ×(63/2)
	00010	31	2.9058×3.2293	2.6013×2.1028	63	(31/2) × (2.9058×3.2293) ×2.6013×2.1028×(63/2)
	000011	31	2.9058×3.2293	NO	63	(31/2) ×(2.9058×3.2293) ×(63/2)
	000010	31	NO	2.6013×2.1028	63	(31/2) × 2.6013×2.1028×(63/2)

Table 4.3: The average number of times to decode one macroblock in a P- or a B-picture with a certain **macroblock_type** parameter.

Accordingly, the average time to guess the headers of all non-intra macroblocks in a P-picture correctly is given by:

$$AverageTime_{(P-picture)} = (1080) \times t_{Fmb} \times r_f \times \frac{L \times W}{16 \times 16} seconds = 4.21875 \times t_{Fmb} \times r_f \times L \times W \; seconds \quad (4.6)$$

where t_{Fmb} is the time of decoding a Forward_Macroblock, r_f is the ratio of macroblocks in a P-picture that are Forward_Macroblocks, and $L \times W$ is the size of the picture. For an MPEG video with frame size of 352×240 and 75% P-Macroblocks in a P-picture and assuming each macroblock requires $\frac{1}{60 \times 330}$ second of decoding, the average time to decode a complete picture is 13.5 seconds.

Similarly, for a non-intra macroblock in a B-picture, the **macroblock_type** parameter can have nine code words as shown in Table (4.3). By assuming that these code words are equiprobable, the average number of times the eavesdropper has to decode a non-intra macroblock in a B-picture is given by summing the values for the B-picture in the sixth column of Table (4.3) and divides it by nine to give *3830* times.

Accordingly, the average time to guess the headers of all non-intra macroblocks in a P-picture correctly is given by:

$$AverageTime_{(B-picture)}$$

$$= (3830) \times [t_{Fmb} \times r_f + t_{Bmb} \times r_b + t_{\mathrm{Im}b} \times r_i] \times \frac{L \times W}{16 \times 16} seconds \qquad (4.7)$$
$$= 14.96 \times [t_{Fmb} \times r_f + t_{Bmb} \times r_b + t_{\mathrm{Im}b} \times r_i] \times L \times W \ seconds$$

where t_{Fmb} is the time in seconds of decoding a Forward_Macroblock, t_{Bmb} is the time in seconds of decoding a Backward_Macroblock, $t_{\mathrm{Im}b}$ is the time in seconds of decoding a Interpolated_Macroblock, r_f is the ratio of Forward_Macroblocks in a B-picture, r_b is the ratio of Backward_Macroblocks in a B-picture, r_I is the ratio of Interpolated_Macroblocks in a B-picture and $L \times W$ is the size of the picture. The time to decode the Forward-Macroblock or the Backward_Macroblock is the same. However, for the Interpolated_Macroblock, this time is more than twice the time needed to decode a Forward_Macroblock because the interpolation process requires some time. So, the following equalities might be valid ones:

$$t_{Fmb} = t_{Bmb} \text{ and } t_{\mathrm{Im}b} = 2.5 \times t_{Fmb} \qquad (4.8)$$

Hence, the average time that is required to guess the header of non-intra macroblocks in one B-picture is given next by equation (4.9).

$$AverageTime_{(B-picture)}$$
$$= 14.96 \times [r_f + r_b + r_i] \times (4.5 t_{Fmb}) \times L \times W \ seconds \qquad (4.9)$$

For an MPEG video with frame size of 352×240, $(r_f + r_b + r_i = 75\%)$ and assuming $t_{Fmb} = \frac{1}{60 \times 330} second$, the time to decode a complete picture is 215.4 *seconds*. In such a

video, a GOP that contains fifteen pictures, four are P-pictures and ten are B-pictures, the average time to guess the headers correctly in all these fourteen pictures and hence get the motion in the video is 1.53 *hours*.

In the method of encrypting 64 bits from every other macroblock regardless of its type, the **Macroblock_address_increment** parameter and the **Macroblock_type** are not known to him and hence he will try all combinations of all the parameters in Table (4.2). Hence, the average number of times the eavesdropper has to decode a non-intra macroblock in a P-or a B-picture is given by equation (4.10).

$$\begin{aligned} AverageTime_{one\ macroblock} &= \frac{33}{2} \times \frac{31}{2} \times 2.9058 \times 3.2293 \times 2.6013 \times 2.1028 \times \frac{63}{2}\ times \\ &= 413512.5716\ times \end{aligned} \tag{4.10}$$

Since, the eavesdropper can not recognize the I-macroblocks from the non-intra ones, he will assume that all the macroblocks in the picture as non-intra. Accordingly, the average time to guess all the headers of the non-intra macroblocks correctly in one picture is given by:

$$\begin{aligned} AverageTime_{(one\ picture)} &= (413512.5716) \times t_{mb} \times \frac{L \times W}{16 \times 16}\ seconds \\ &= 1615.283 \times t_{mb} \times L \times W\ seconds \end{aligned} \tag{4.11}$$

where t_{mb} is the time in seconds of decoding a Macroblock assuming that all Macroblocks require the same time for decoding.

For an MPEG video with frame size of 352×240 and assuming each macroblock requires $\frac{1}{60 \times 330} second$ of decoding, the average time to decode a complete picture is 1.914 *hours*. In such a video, a GOP that contains fifteen pictures, four are P-pictures and ten are B-pictures, the average time to guess the headers correctly in all these fourteen pictures and hence get the motion in the video is 26.802 *hours*.

It is very clear that the average time required for guessing one B-picture is larger than that required for guessing a P-picture. This is expected because of more parameters be involved in calculating the combinations of the macroblock header. If the eavesdropper manages to guess all the headers of the non-intra macroblocks, he still has to decrypt the intra-macroblocks. So, the maximum the eavesdropper will gain by using this attack against the proposed secure system of either encrypting the non-intra macroblocks' headers as well as all intra-macroblocks or encrypting 64-bit segment from every other macroblock is to have an MPEG video with only the intra-macroblocks are lost. Also, the method of encrypting 64-bit segment from every other macroblock enhances the security level and reduces the required processing time compared to the other methods.

4.2.2 Bit stream Resynchronization

The length of the header of a non-intra macroblock can be between 2 and 121 bits while for an I-MB, it can be between 2 and 53 bits. So, encrypting only a 64-bit segment of the header either includes part of the DCT-coefficients or excludes some header parameters. In both cases, this method introduces a synchronization problem for the eavesdropper in decoding the

unencrypted part of the non-intra macroblock. Also, this synchronization problem exists in the method of encrypting I-MB's where some bits, that are less than 64 bits, are left unencrypted and the eavesdropper will try to synchronize. Assume that the eavesdropper knows that the part of the MPEG-I video bit stream being selectively encrypted. At a certain position in the bit stream the eavesdropper will try to decode a VLC. If he can not decode the macroblock correctly, then he will realize that he is not at the start of a codeword. Consequently, he will shift the bit stream by one bit and attempt decoding once again. The eavesdropper will repeat this process until he regains synchronization. Decoding a block in the MPEG-I bit stream is shown in Figure (4.2).

There are two parameters in calculating the average time that is required to resynchronize. The first one is the number of times the eavesdropper decodes the macroblock at each time he shifts and the second parameter is the number of shifts he has to perform before synchronization is achieved. It is not possible to find the time of synchronization by using the technique used for self-synchronizing VLC's because the VLC's used in MPEG-I do not have synchronizing codewords [17]. Let us first find the number of times the eavesdropper has to decode the macroblock before shifting.

The time to resynchronize depends on the assumptions and guesses the eavesdropper will perform. These guesses depend on the type of the macroblock whether it is intra or not. Hence, first the time to resynchronize will be calculated when the macroblock is non-intra and then when the macroblock is intra.

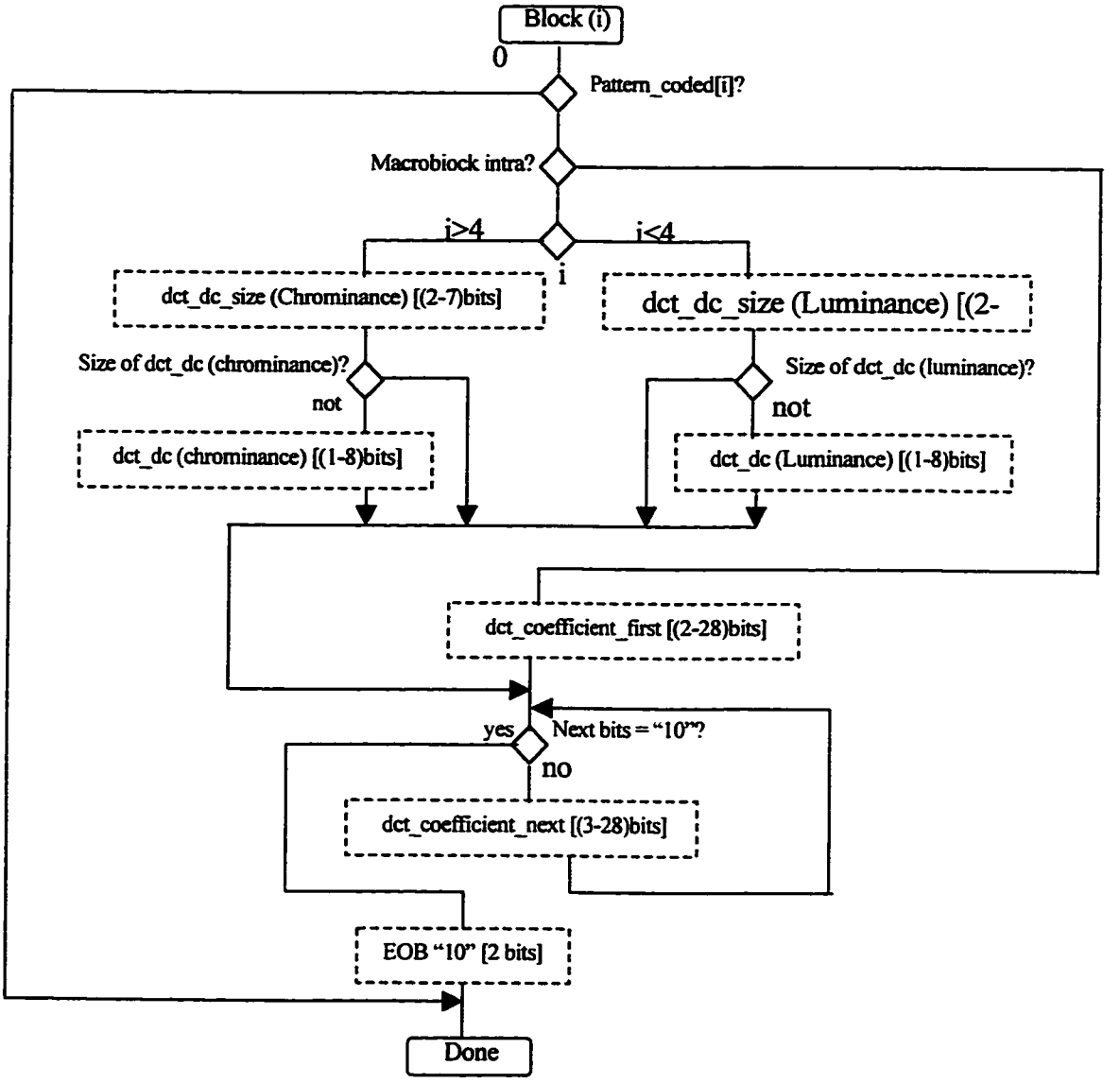


Figure 4.2: MPEG-1 Block Decoding Loop.

At the end of the encrypted segment in a non-intra macroblock the eavesdropper will try to resynchronize. He has two possible guesses. He either guesses the next code word as a first-dct coefficient or as next-dct coefficient as shown in Figure (4.2). The list of VLC's that are valid code words for first-dct coefficient includes 224 code words while the list for the next-dct coefficient includes 222 code words [17]. Both lists include also the *escape* code word (000001). When the decoder finds this code, it reads the next 8 bits. If these 8 bits are not between -127 and 127, it reads the next 8 bits to have 16 bits that determine the level of the DCT-coefficient. Hence, the total number of code words the *escape* code words represents is 512 values [17]. Accordingly, the maximum number of times the eavesdropper has to decode the non-intra macroblock at each shift is (512+222+224) = 958 times.

If the macroblock is intra and the eavesdropper is at a certain position in the macroblock unencrypted bit stream, he has to guess one among three possibilities. These possibilities for the current code word are as follows: it represents the DC_dct coefficient of a luminance block, it represents the DC_dct coefficient of a chrominance block and it represents the AC_dct coefficient.

The DC coefficient is encoded differentially and two codes are used as shown in Figure (4.2). One, 1-to-8 bits, gives the size of the fixed code. The other is the fixed code itself that represents the differential DC coefficient. Hence, guessing the current code word to be DC_dct_luminance/chrominance results in $(2^1 + 2^2 + 2^3 + ... + 2^7 + 2^8)$ 510 times of decoding.

The AC_dct_coefficient is encoded using 222 VLC's and the *escape* code, which represents 512 different code words [17]. Hence, if the eavesdropper guesses the current code word to represent AC_dct_coefficient, then he has to decode the macroblock 734 times.

As a result, for the four possibilities the total number of times the eavesdropper has to decode the I-MB to resynchronize in decoding the unencrypted VLC's in an I-MB is: $510 \times 2 + (222 + 512) = 1754$ times. The *2* factor is to account for the two kinds of blocks; i.e. luminance and chrominance.

The next step in calculating the average time that is required to resynchronize in decoding one intra or non-intra macroblock is to find the number of shifts. There are two scenarios.

1. The header is less than 64-bit in length. This case is valid whether the macroblock is intra or non-intra. Hence, the first bit after the encrypted segment can be the first bit, the last bit or any bit of a valid codeword. Since the maximum length of a codeword is 28 bits, then the maximum number of shifts in this case is 28 shifts. So, the average number of times to decode a non-intra macroblock is $\frac{28 \times 958}{2} times$. Therefore, the average time to resynchronize in a non-intra macroblock is:

$$\text{Average time to synchronize for one non-intra macroblock} = 13412 \times t_{mb} second \quad (4.12)$$

Where, t_{mb} is the time required to decode one macroblock in seconds. If this time is $\frac{1}{60\times330}second$, then the average time to synchronize for one non-intra macroblock is 0.68 seconds.

However, the average number of times to decode an intra macroblock is $\frac{28\times1754}{2}times$.

Therefore, the average time to resynchronize in an intra macroblock is:

Average time to synchronize for one I-MB = $24556\times t_{mb}second$ (4.13)

Where, t_{mb} is the time required to decode one I-MB in seconds. If this time is $\frac{1}{60\times330}second$, then the average time to synchronize for one non-intra macroblock is 1.24 seconds.

2. The header is more than 64-bit in length. This case is valid only in non-intra macroblocks because the length of the header of an I-MB will never reach 64 bits. So, for a non-intra macroblock, the first bit after the 64-bit segment is not part of a valid codeword. Since the maximum length of the header is 121 bits, then the maximum number of shifts in this case is (121-64) = 57 shifts. So, the average number of times to decode the non-intra macroblock is $\frac{57\times958}{2}times$. Therefore:

Average time to synchronize for one macroblock = $27303\times t_{mb}second$ (4.14)

Where, t_{mb} is the time required to decode one non-intra macroblock in seconds. If his time is $\frac{1}{60\times 330}\text{sec}ond$, then the average time to synchronize for one non-intra macroblock is 1.38 seconds.

By comparing equations 4.1-4.8 one notices that the required time to resynchronize is negligible compared to the time of guessing correctly the headers. Also, the gain of synchronization in I-MB's is not much because the DC components are encoded differentially and losing these components implies losing the whole block even if the AC components are present. Hence, the maximum the eavesdropper can do is to guess the header to get the motion in the video clip but this requires long time as shown above.

In the first scenario the part of the DCT-coefficients which is encrypted can be the low frequency DCT-coefficients or it can be one or more blocks and the low frequency DCT-coefficients of the subsequent block. Since, these low frequency coefficients contain most of the information of the block then losing these coefficients causes a loss of the block. If the eavesdropper manages to regain bit stream synchronization, the high order coefficients will substitute the encrypted low order coefficients. Unless these coefficients are shifted back to their correct position the image quality is not acceptable. This repositioning can be achieved by trial and error. Even though the decoded P-or the B-picture will still have the intra coded macroblocks encrypted.

The effect of shifting the DCT coefficients is simulated and the results are shown in Figure (4.3). In this figure (a) is the original image, (b) is the resulting image after shifting the coefficients of the first block, (c) is the resulting image after lossing the first block and shifting the coefficients of the second block and (d) is the resulting image after loosing the first and the second block and shifting the coefficients of the third block.In the first experiment, the DCT-coefficients of the first block in each macroblock are shift down two positions. The result of this experiment is shown in Figure (4.3-b). The second experiment illustrates the case when the 64-bit segment being encrypted includes the header, and the coefficients of one or more full blocks and the low-frequency components of the next block. Hence, the fully encrypted blocks were eliminated and the DCT-coefficients of the partially encrypted block are shifted down two positions. Figure (4.3-c) shows the results when only the first luminance block is fully encrypted. Figure (4.3-d) simulates the results when the first and the second blocks are fully encrypted. From these figures, it is clear that losing the low-frequency components of a block causes sever degradation. Hence, even after performing this guessing by the eavesdropper for the re-synchronization, still guessing the headers correctly requires long time and both the method of encrypting all I-MB's as well as the headers of the non-intra macroblocks and the method of encrypting 64-bit segment from every other macroblock remain secure against the attacks discussed so far.

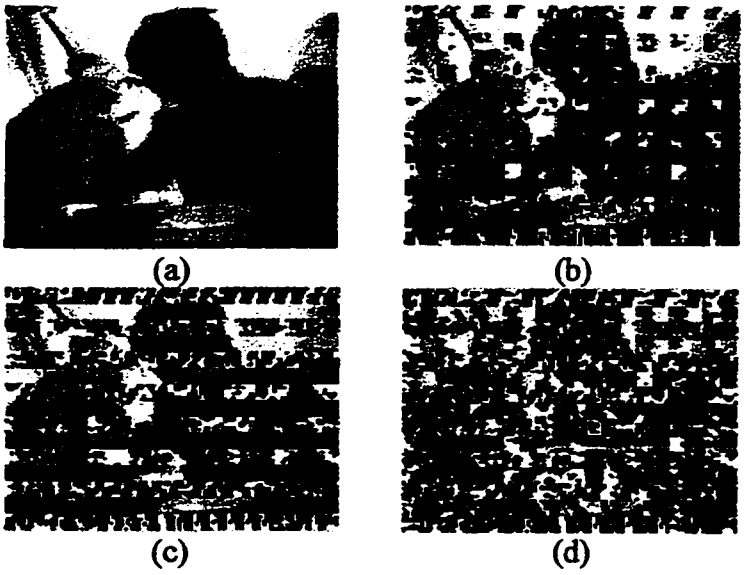


Figure 4.3: The effect of shifting down the highest two DCT-coefficients of the blocks in a macroblock.

4.3 Transmitting the DES Key within MPEG-I video bit stream

The process key, which is the DES key used to encrypt/decrypt the data in the MPEG-I video bit stream, has to be exchanged between the transmitter and the receiver. This is because the key has to be changed periodically as will be shown in section 4.4. If the cryptanalyst attacks the system and he gets the correct key used to encrypt/decrypt a certain number of GOP's then changing the key periodically will not allow him to compromise the whole video. For the reader to evaluate this, he might calculate how frequently he changes his UNIX account password or his bankcard password. For very sensitive MPEG video applications the key is changed periodically within transmission. So, the classical methods of communicating the key such as a messenger or a special channel are not applicable. Thus, the key has to be transmitted within the MPEG video bit stream. However, its location in the bit stream should

not disturb the syntax of the bit stream and it should be easy for the receiver to extract it. In this thesis, the proposed location of the process key is in the USER_DATA part in the picture header in MPEG-I video bit stream as shown in Figure (4.4). This location will not disturb the bit stream. Furthermore, it will not cause any delay or storage requirement at the receiver because at the time of getting the process-key non-of the so far received data is encrypted by this key. Transmitting the process key this way introduces overhead. This overhead is 112 bits if the process key is sent unencrypted. 32 bits are for the USER_DATA_START_CODE (0x000001B2), 24 bits are inserted after the user data information to tell the receiver that the user data bits have finished as shown in Figure (4.4)Figure. The remaining 56 bits are for the DES key. This overhead is tolerable if the key is not changed so frequently. The upper extreme is to change the key at each frame. The overhead in this case is 1320 bits in one GOP that has 15 frames. On the other hand, the low extreme is to have one key for the whole MPEG video sequence and hence the overhead in all the video sequence is just 112 bits.

```
If (nextbits == 0x000001B2){
        User_Data_Start_Code
        While ( nextbits NOT EQUAL '0000 0000 0000 0000 0000 0001'){
                User_Data
        }
        next_start_code()
}
```

Figure 4.4: The syntax in the Picture header to search for any user data sent within MPEG-I video bit stream.

The only way to secure the process key while it is transmitted within the MPEG video bit stream is to encrypt it before transmitting it. A method to do that is to use a public-key cryptosystem such as the RSA to encrypt/decrypt the DES key. This method has the

advantage that the cryptanalyst can not apply the same method to attack the two keys because the way of attacking the private key algorithm is different than attacking the public-key algorithm as will be seen in the next section. The method of incorporating both symmetric and asymmetric cryptosystems enhances the security level considerably [10]. So, the transmitter will use the public-key *e* of the RSA to encrypt the process key (the DES key). Accordingly, the intended receiver is the only authorized one who can decrypt the process key using his private-key *d*. Hence, he can use the process key (the DES key) to decrypt the encrypted part of the MPEG bit stream. The RSA is not used to encrypt/decrypt the data in the MPEG-I bit stream because of the long processing time [10] it requires as explained in chapter one of this thesis. RSA has been chosen among various public-key algorithms because it is the best known and the most powerful one [10].

4.4 Cryptanalyst's Attacks on DES and RSA Keys

Having a very powerful crypto-algorithm that produces an output, which does not convey any information about the input, does not guarantee security without protecting the keys [31, 10]. The cryptanalyst can attack the system by applying certain methods. He can work on finding the process key (the DES key) used to encrypt/decrypt the MPEG video bit stream. Alternatively, he might attack the system by trying to find the private/public keys of the RSA algorithm used to protect the process key (the DES key). The third option for the cryptanalyst is to attack both keys, the DES and the RSA-private keys, simultaneously. This requires very high computational power and high cost because each attack will have its own method. This

is another advantage of using RSA in protecting the DES key. An analysis of each attack is explained separately in the following sub-sections. This analysis gives an estimation of the lifetime of the keys and an estimation of the attack cost.

4.4.1 Brute-Force Attack on The Process Key (The DES Key)

To attack the DES key, the cryptanalyst can apply two methods to compromise the process key (the DES key). These methods are the Brute-Force attack and the differential analysis attack. The former is a known-plaintext attack while the latter is a chosen-plaintext attack [10]. The main idea of the differential analysis attack is to choose an input pairs, to the DES algorithm, that have certain difference and notice the difference in the resulting output pairs. Repeating this scheme for large number of pairs will increase the chance of emerging to the right key being used. Each input difference produces an output difference with some probability. More detailed description of the differential analysis attack is shown in [10,32]. However, in this thesis, this attack is not chosen to evaluate the security level of the system because this attack is largely theoretical, impractical and it requires a huge part of the encrypted bits [10]. Biham and Shamir [32] have used clever mathematics to achieve, theoretically, a successful attack against the full 16-round DES. Their attack is considered to be the best attack of differential analysis type. However, their attack required 2^{55} known plaintext (2^{55} bits as input to the DES algorithm), 2^{47} chosen plaintext and 2^{37} DES operations [10,32]. Furthermore, the 2^{55} input bits have to be all encrypted using the same key. If the cryptographer keeps changing the key frequently, there will not be

available 2^{55} bits (36,028,797,018,960,000 bits or equivalently 562,949,953,421,300 DES-blocks) that are encrypted with the same key for the differential cryptanalyst. The required known-plaintext for the differential analysis to proceed is 2^{55} bits. In MPEG video sequences I-frames contain more data than non-intra frames. By calculating the average number of bits in one I-frame and assuming that all frames in the video sequence to be intra, we can calculate the number of frames that have together 2^{55} bits. Accordingly this number of frames gives us how frequently the key has to be changed. This is calculated for all the test video clips used in this thesis in Table (4.4). The assumption is that all frames are of type intra.

Name of the MPEG video sequence	**Average number of bits in an intra-frame**	**Maximum number of frames that can be encrypted with the same key**
Fibon	132953	2.7×10^{11}
Para	89971	4.0×10^{11}
Creamedgates	58614	6.1×10^{11}
Gulf_war	90602	3.98×10^{11}
Kangaroo	83026	4.3×10^{11}
Fish	56432	6.4×10^{11}
Falcon	105248	3.42×10^{11}
Oj_Chase	90182	4.0×10^{11}

Table 4.4: The maximum number of frames at which the same key can be used to encrypt all frames without having a threat from differential cryptanalysts. (Assuming all frames are of type intra.)

From the table, the maximum number of frames that can be encrypted with the same key is in the order of 10^{11}. So, a 105-years MPEG video sequence can be encrypted thoroughly without being successfully attacked by the current differential analysis attack techniques.

Furthermore, in this thesis the selective encryption/decryption method is applied and not all frames are intra. This implies that there is no threat from the current differential analysis attack. The word "current" is used here because in the future there might be advances in applying the differential analysis technique that might be more efficient than the one in [32], in minimizing the required known plaintext size.

The other attack the cryptanalyst can apply is the Brute-Force attack. It is a known-plaintext attack. It requires two 64-bit DES-blocks that represent the input and the corresponding output of the DES algorithm. Furthermore, MPEG video bit stream has standard headers and its syntax is available for the public in the literature. This makes the Brute-Force attack appropriate to attack the secure transmission of MPEG video sequences. The job of the cryptanalyst is to try all the 2^{56} possible keys to find the right key. The cryptanalyst has to know some information before starting his Brute-force attack against the selective encryption/decryption methods proposed in this thesis. First of all, he has to know what parts of the video clip are encrypted. Second, he has to know the algorithms used in securing the MPEG video sequence. Such information can be obtained easily through insiders. So, in this thesis we assume that the cryptanalyst is capable of knowing what parts are encrypted and also he can listen successfully to the communication channel between the transmitter and the receiver.

In chapter 1, a general discussion of the Brute-Force attack has been presented. Time and cost calculations have been conducted for different key lengths. The equations used are reproduced here for convenience.

$$Time = \frac{2^k}{L \times n \times 3600} hours \tag{4.14}$$

$$Cost = n \times c \tag{4.15}$$

where:

n : # of machines used,
L : # of keys tested per second by one machine,
k : key length in bits, and
c : cost of each machine.

The estimated time and cost for the process key (the DES key) that has 56 bits is reproduced in Table (4.5).

Number of used machines	Cost (U.S. $)	Time spent to get the DES key
1	C	(2.3/L) G-years
1000	1000×c	(2.3/L) M-years
10,000	10,000×c	(230/L) K-years
100,000	100,000×c	(23/L) K-years
1,000,000	1,000,000×c	(2.3/L) K-years

Table 4.5: Time and cost needed to identify the DES key by Brute-Force attack for a machine that can test *L* keys/sec; *c* is the cost of one machine.

Some keys require testing time larger than others. However, we will assume that all keys have the same testing time [10]. The variable *L*, which is the machine capability of testing a key and the variable *c* is the cost of one machine. To have practical calculations, both *L* and *c* must have certain values. These values can be obtained either from the state-of-the-art workstations that can test certain number of keys per second or from previous conducted Brute-Force attack calculations on the DES algorithm.

Shneier in [10] mentioned that a researcher in 1991 conducted Brute-Force attack calculations. The researcher assumed the availability of a machine that can test 8192 keys in one second. Moore's law states that *computing power doubles approximately every 18 months* and accordingly the cost goes down a factor of 2 every year [10]. At the time of writing this thesis, 1998, the machine capability according to Moore's law becomes 76460 keys in one second. So, substituting for $L = 76460$ in Table (4.5) gives the calculations in Table (4.6). We all assume that the cost of one machine is U.S. $500.

Number of used machines	**Cost (U.S. $)**	**Time spent to get the DES key**
1	500	30081 years
1000	5×10^5	30 years
10,000	5 Million	3 years
100,000	50 Millions	110 days
1,000,000	500 Millions	11 days

Table 4.6: Time and cost needed to identify the DES key by Brute-Force attack for a machine that can test 76460 keys/sec; the cost of one machine is U.S. $500.

The other source for the L values is from the state-of-the-art machines. If we assume that one state-of-the-art machines can test 10^6 keys in one second, then the calculations become as in Table (4.7). The cost of one machine is assumed to be U.S. $ 800. On the same table, similar calculations are made if the machine can test 10^9 keys in one second. However, the cost of such a machine is assumed to be U.S. $1000. So, the worst case for the cryptographer (the best for the cryptanalyst) is to find the key in maximum time of 72 seconds. This requires million machines working 24 hours a day.

To estimate the required time of changing the key, let us take the worst case, which is in Table (4.7). The cryptanalyst might get the key in 72 seconds. By taking the average, the cryptographer has to change the DES key each 36 seconds. To translate this into umber of frames, the cryptographer has to change the key each 1050 frames. This is if the transmission rate is 30 fps. Taking a safety factor of 20%, then the key has to be changed every 800-frame. If each GOP has 15 frames, then the key is updated every 53 GOP's. For *fibon* video clip, which has 6773 frames, secure transmission requires 9 DES keys. The overhead caused by transmitting 9 keys is negligible and it does not affect the transmission rate.

	One machine can test 10^6 keys in one second		***One machine can test 10^9 keys in one second***	
Number of used machines	**Cost (U.S. $)**	**Time spent to get the DES key**	**Cost (U.S. $)**	**Time spent to get the DES key**
1	800	2300 years	1000	834 days
1000	8×10^5	2 years	1 Million	20 hours
10,000	8 Millions	84 days	2 Million	2 hours
100,000	80 Millions	8 days	100 Millions	12 minutes
1,000,000	800 Millions	19 hours	1 Billion	72 seconds

Table 4.7: Time and cost needed to identify the DES key by Brute-Force attack for two different state-of-the-art machines.

4.4.2 Attacking the RSA Key

Another alternative for the cryptanalyst to attack the secure system, proposed in this thesis, is to find the RSA-private key d to decrypt the process key (the DES key) that exists in the user-data part. Consequently, he can decrypt the encrypted part of the MPEG bit stream. To do so, he must know the modulus n and the public key e. Assuming the cryptanalyst knows both n

and e, he will try to find the private key d from these two quantities. It has been found that finding d from e is as hard as factoring n to get the primes, p and q [10,33]. Even though a high improvement in the computation technology and the number theory has been achieved since 1978, when RSA was proposed, no method has been found that is more obvious and more efficient than factoring n to attack the RSA. Even the Brute-Force attack is considered to be less efficient than factoring the modulus n in attacking the RSA algorithm [10]. Furthermore, the difficulty of factoring large integers has not been proved but no method that factors such integers easily has been introduced. This need for factorization has opened a good research area for mathematicians to work it out.

The most efficient and the most practical algorithms used to factorize large integers are called the Quadratic Sieve (QS) and the General Number Field Sieve (GNFS). These schemes suits perfectly for the parallel implementation of the attack [10]. So, the cryptanalyst can make many machines work on finding the key. These factoring algorithms require both high speed CPU's and large memory [40]. The mathematics behind these two algorithms is beyond the scope of the discussion in this thesis but what concerns us is how long it will take the cryptanalyst to factor the large integer using one of these two algorithms. The researchers estimate the complexity of factoring a specific integer from previous successful factorizations of integers with shorter lengths [40]. There are general assumptions that can be applied to help in estimating both time and cost of factorization. These assumptions are:

1. In every decade, one can factor numbers, which are twice as long as those factored in the last decade [10].

2. For numbers around 100-digit adding 3 digits doubles the computed time needed for QS [38].

3. The computing power doubles every 18 months. This is called Moore's law [10,39,40].

All previous successes in factorizing RSA-numbers have been achieved using the QS algorithm. However, the GNFS is relatively new, introduced in 1991, and is more promising [10].

Computing power is measured in MIPS-years: one-Million-Instructions-Per-Second. It represents the number of operations completed in a year by a machine operating at one million instructions per second [10, 39]. To choose an RSA key-length for the system proposed in this thesis, all previous successes in factoring integer lengths have to be studied with the applicability of the assumptions listed above. Table (4.8) lists the lengths of the integers being factored. Also, it gives the year and the computation power each length required to be factored.

The first assumption in the list made previously is valid because between 1974 and 1984 and between 1984 and 1994, the integer being factored has, nearly, doubled. Similarly, according to the second assumption and since the RSA-116 required 400 MIPS-years, then factoring the RSA-120 number is expected to require 800-900 MIPS-years. As can be seen from Table (4.8), RSA-120 took 825 MIPS-years. However, the second assumption can not be used to estimate the requirements of the RSA-129 from the RSA-120 requirements because this assumption is valid only for number s around 100-digit.

Integer Length (in digits)	Year of Successful Factorization	Time/Computation Power Required
45	1974	0.001 MIPS-years
71	1984	0.1 MIPS-years
100	1988	1 month
116	1991	400 MIPS-years
120	1993	825 MIPS-years (3 months)
129	1994	5000 MIPS-years (8months and 1600 machines)

Table 4.8: Historical records of the successful large integer factorizations.

The most recent factorization is that achieved by Atkin and others, when they factored the RSA-129 number in 1994 [10]. They applied the QS algorithm. It required 5000 MIPS-years. They used 1600 machines around the world to achieve the factorization. So, the RSA-129 number is not secure any more. The next number is a 512-bit long. Does it provide enough security these days? Or we must use longer numbers. To answer these questions, we should do some calculations according to the previous factorizations and the three assumptions mentioned before.

Robshaw in [39] had estimated the security of the 512-bit RSA modulus. He assumed the validity of 10 MIPS-years machines with cost of U.S. $500 per one machine. Since he made these assumptions in 1995, and the time of writing this thesis is 1998, then the machine capability has improved over the three years. According to the third assumption listed earlier in this section, we will assume the availability of 40 MIPS-years machines with a cost of U.S. $500 per one machine. In here the improvement in the technology is reflected in the machine capability and hence the price remains the same.

Odlyzko had estimated in [40] that factorization of the 512-bit integer requires 3×10^{9} MIPS-years. He also estimated the run time required to factor different RSA modulus lengths. His estimations are listed in Table (4.9). The basis in these estimations is that the RSA-119 required 250 MIPS-years. By projection the estimated computation requirements for longer integers have been found to be very close to what they require in reality [40]

RSA-modulus Length (in bits)	Expected MIPS-years Required for Factoring
512	3×10^{9}
768	2×10^{8}
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

Table 4.9: The MIPS-years requirements to factor different RSA-modulus lengths as estimated by Odlyzko [40].

According to these estimations and the assumptions that have been made earlier on the machines available, the time and cost of factoring RSA-modulus integers with different lengths are in Table (4.10). Also, the table estimates these requirements in the coming six years. The equation used is:

$$Time = \frac{OPER_{req.}}{\left(\frac{Total\ Cost}{Cost\ of\ One\ Machine}\right) \times OPER_{available}}$$

where:

$OPER_{req}$ is the number of operations, in MIPS-years, required to factor an integer with certain length.

$OPER_{availabl}$ is the number of operations that can be completed by one machine.

If a cryptanalyst is powerful enough to have the technology of the year 2000 nowadays, then he can factor the 512-bit RSA-modulus in 12 hours while the 768-bit RSA-modulus in 10 years. If the cryptographer is transmitting an MPEG video sequence that consists of two Million frames (18 hours video sequence) then the cryptanalyst can factor the 512-bit RSA-modulus and consequently get the private key before the end of the transmission.

Choosing an RSA-modulus of length 1024 bits other than 678-bit implies a larger time for the RSA encryption/decryption processes. Schneier in [10] gave an estimation of the encryption/decryption processes time. This time is estimated to be 1.01 second when using the 1024-bit RSA-modulus. In the last section we decided to change the DES key each 800 frames. So, using the 1024-bit RSA-modulus to encrypt/decrypt the DES key will decrease the frame rate to 28.9 fps. However, using the 768-bit RSA-modulus will degrade the transmission rate to be 29.4 fps.

So, if the transmission time of an MPEG video sequence secured by one of the methods discussed in chapter three is less than twelve hours, then using the RSA-512 key is secure. However, if the transmission time is expected to be more than twelve hours then the RSA-768 key is preferable.

		RSA-modulus length (in bits)		
Total Cost (U.S. $)	**Year**	**512**	**768**	**1024**
100,000	1998	3.8 years	25000 years	37500000 years
	2000	1.5 years	9900 years	14851485 years
	2002	7 months	3937 years	5905512 years
	2004	2.8 month	1563 years	2343750 years
1,000,000	1998	4.5 months	2500 years	3750000 years
	2000	1.8 months	991 years	1485149 years
	2002	22 days	394 years	590552 years
	2004	9 days	157 years	234375 years
10,000,000	1998	14 days	250 years	375000 years
	2000	5 days	99 years	148515 years
	2002	2 days	39 years	59055 years
	2004	22 hours	16 years	23438 years
100,000,000	1998	34 hours	25 years	37500 years
	2000	12 hours	10 years	14852 years
	2002	5 hours	4 years	5906 years
	2004	132 minutes	2 years	2344 years

Table 4.10: The estimated time and cost required for factoring an RSA-modulus with different lengths and in different years. (The cost of one machine is U.S. $500).

4.5 Summary

In this chapter, different procedures that can be used by an eavesdropper to compromise the secure system have been studied. First of all, a procedure to attack the method of encrypting the headers of non-intra macroblocks through guessing has been investigated. This procedure depends mainly on guessing the header parameters for each macroblock and then resynchronizes to the first unencrypted DCT-coefficient. It has been concluded that this attack requires considerable time and the resulting video will have the effect of encrypting the I-MB's and hence only the motion will be revealed. So, for real-time applications this attack fails in compromising the secured system.

On the other hand, attacking the DES and the RSA keys through cryptanalysis costs money and it depends mainly on the machine used while the algorithm is known. According to Table (4.7) and Table (4.10), for 100 Million US Dollars, the cryptanalyst can get the DES key in 12 minutes and the 768-bit RSA key in 25 years. It has been concluded that the DES key has to be changed every 800 frames.

The guessing method provides the eavesdropper with the nature of the motion in the MPEG video and some numbers or words if they exist in the video as shown for the *fibon* video clip. On the other hand, attacking the key will provide the cryptanalyst completely with the original video frames that are encrypted with the same key. The cryptanalysis attack costs money as shown in the last section. So, the choice between the two attacks depends on the resources the eavesdropper has, the importance of the video and on whether it is enough to

know the motion in the video without details or not. So, if the eavesdropper wants to know the nature of the motion in the MPEG video, he can implement the guessing method without losing money. If he cares about the details in the video, he has to go to cryptanalysis attacks.

Chapter 5

Conclusions and Future Research

5.1 Conclusions and summary

In this thesis, we have studied several techniques to selectively encrypt MPEG-I video bit streams. Different selective techniques have been suggested in the literature. These techniques include encryption I-pictures. These techniques have been evaluated and the results show the lack in the security level in the first method while the second method requires large processing time. Many suggestions to improve the level of security have been evaluated. These suggestions include increasing the I-frame frequency [5, 18] and encrypting both I- and P-frames [5] in the video clip. Although, the security level is improved by these

solutions, they diverge from the main idea of selective encryption. Also, classifying P-frames as part of the independent data is not valid because they depend on I-frames. In addition, τ resulting from the method of increasing I-frame frequency becomes more than 0.50 without disguising the video completely. In the case of encrypting both I- and P-frames, τ becomes very close to 1, which diverges from the purpose of selective encryption.

Four methods have been proposed in this thesis to improve the security level by keeping the required processing time small. The first method calls to consider all I-macroblocks in all frames as the independent part. This method of encrypting all I-MB's in all frames has been tested and compared to the method of encrypting I-frames only. The experimental results show an improvement in the security level. However, the motion is still clearly recognized and the processing time increases. This increase in the processing time or, equivalently, this degradation in the processing time depends on how many macroblocks in P- and B-frames are encoded as intra and on how many bits these macroblocks contain. This method has been improved further by encrypting half the I-MB's. The security level remains almost the same, while the processing time has been reduced by, nearly, half.

The content of the MPEG-I video clips encrypted by this method is disguised except for the motion, where one can recognize the motion of a body in the encrypted video clip. This motion information exists in the headers of the non-intra macroblocks that exist in the non-intra frames (P and B). So, disguising this information will remove the motion information from the bit-stream and hence improves the security level. This led to the second method of the MPEG video sequence. This method calls to consider, in addition to the I-MB's, the

headers of the non-intra macroblocks as the independent part of MPEG-I video bit-stream. This is because of the existence of the motion information in this part of the MPEG video sequence. So, the motion information has been disguised by further encrypting the headers of P- and B-macroblocks in addition to all I-macroblocks in all frames. The results of this method show a completely disguised transmission of MPEG video sequences. However, the processing time increases compared to encrypting I-macroblocks only. The fraction of the video data being encrypted, τ , achieved by encrypting I-macroblocks only is between 0.23 and 0.35 while by further encrypting the headers of P- and B- macroblocks the fraction of the video data being encrypted is between 0.36 and 0.79.

This method has been improved by encrypting half the I-MB's and all the headers of the non-intra macroblocks. This is the third method proposed in this thesis and it gets use of the DPCM coding technique by applying alternative encryption. The security level remains the same while the fraction of the video data being encrypted, τ , decreases to be between 0.23 and 0.63. By encrypting half the I-MB's and the headers of half the non-intra macroblocks, the fraction of the video data being encrypted, τ , decreases to be between 0.18 and 0.39. However, the security level remains the same where the MPEG video sequence is disguised completely. Further reduction in the processing time can be achieved by encrypting half the I-MB's and the headers of third the non-intra macroblocks. However, decoding the encrypted video clips using this method show a considerable degradation in the security level.

The fourth method makes use of the predictive DPCM coding of the DC coefficient of the DCT block in I-macroblocks. This method calls to encrypt 64-bit segment from every other

macroblock regardless of its type. This scheme gives fully secure transmission of MPEG video clips while the ratio of the data being encrypted is kept less than 0.29.

All the above results assume that the eavesdropper will try to get use of the unencrypted part of the MPEG video bit stream. However, the proposed methods ensure that the unencrypted part does not reveal information about the video clip. Hence, an analysis of the proposed methods has been conducted. This analysis assumes that the eavesdropper will not get satisfied with the encrypted video clips and he chooses to attack the secure system. Three different attacks have been analyzed; namely guessing the headers of the macroblocks, re-synchronizing in the unencrypted part of the macroblocks data and attacking the encryption keys. This analysis shows that the system remains secure for real-time applications and the cryptanalyst does not have enough time to attack the system.

The Data Encryption Standard (DES) algorithm has been used to encrypt the MPEG bit stream while the RSA algorithm has been used to encrypt the DES key. The encrypted DES key is changed frequently and it is sent to the receiver as user data in the picture header. In this thesis, different procedures that can be used by an eavesdropper to compromise the secure system have been studied. First of all, a procedure to attack the method of encrypting the headers of non-intra macroblocks through guessing has been investigated. This procedure depends mainly on guessing the header parameters for each macroblock and then resynchronizes to the first unencrypted DCT-coefficient. It is concluded that this attack requires considerable time and the resulting video will have the effect of encrypting the I-

MB's and hence only the motion will be revealed. So, for real-time applications this attack fails in compromising the secured system.

On the other hand, attacking the DES and the RSA keys through cryptanalysis costs money and it depends mainly on the machine used while the algorithm is known. According to both Table (4.7) and Table (4.10), for 100 Million US Dollars, the cryptanalyst can get the DES key in 12 minutes and the 768-bit RSA key in 25 years. It has been concluded that the DES key has to be changed every 800 frames.

The guessing method provides the eavesdropper with the nature of the motion in the MPEG video and some numbers or words if they exist in the video clip. On the other hand, attacking the key will provide the cryptanalyst completely with the original video frames that are encrypted with the same key. The cryptanalysis attack costs money as shown in the last section. So, the choice between the two attacks depends on the resources the eavesdropper has, the importance of the video and on whether it is enough to know the motion in the video without details or not. So, if the eavesdropper wants to know the nature of the motion in the MPEG video, he can implement the guessing method without losing money. If he cares about the details in the video, he has to go to cryptanalysis attacks.

5.2 Future Research

The classification methods introduced in this thesis are valid for all members of MPEG family (MPEG-I, MPEG-II and MPEG-IV) because all of them have the same structure of

MPEG-I with some more features at certain layers. An improvement can take place in MPEG-IV where a technique called content-based coding is used to code different areas in the same picture according to their contents. Hence, another classification of MPEG-IV video bit stream may reduce the processing time. Further research is needed to investigate this possibility.

A further study has to take place to investigate the effect of the proposed schemes of selective encryption on the received MPEG video sequences over different network environments such as ATM networks. The study has to investigate the effect of selective encryption on the loss rate over such an environment.

In this thesis we have not considered encrypting the audio. We concentrated on the video part of MPEG. It would be natural to encrypt the audio too if it contains valuable information. Also, different selective encryption techniques have to be discussed for the audio not to cause delay for the video part since they are multiplexed in the system layer of MPEG.

Furthermore, a study has to consider using the proposed techniques in video watermarking. Such techniques can be very useful and efficient to achieve the goal of watermarking.

The results of this thesis can be viewed as a necessary background for future work in this field.

Appendix

In the attached CD-ROM, there are three main folders.

1) VMPEG: This folder has the VMPEG 1.7d-lite decoder that is used in the thesis to decode the encrypted MPEG video clips. It is allowed to distribute it for demonstration purposes only and not for commercial use as mentioned in the ReadMe.txt file.

2) ZEROING: This folder contains a folder for each test video clip. Each folder contains four .mpg files. Each start with a number followed by the name of the video clip (e.g. 2_Fibon.mog). The interpretation of these numbers is:

Z1_:	All I-pictures are zeroed
Z2_:	All I- and P-pictures are zeroed
Z3_:	All I-Mb's in all frames are zeroed
Z4_:	All I-MB's are zeroed and the headers of the non-intra macroblocks are confused

3) ENCRYPTION: In this folder there exists a folder for each test video clip. Each folder contains nine .mpg files. Each start with a number followed by the name of the video clip (e.g. 2_Fibon.mog). Also, each folder contains the original video clip. The interpretation of these numbers is:

E1_:	Encrypting all I-picture
E2_:	Encrypting I- and P-pictures
E3_:	Encrypting all I-MB's
E4_:	Encrypting all I-MB's and the headers of the non-intra macroblocks
E5_:	Encrypting I-MB's alternatively
E6_:	Encrypting I-MB's alternatively and all the headers of the non-intra macroblocks
E7_:	Encrypting I-MB's alternatively and the headers of the non-intra macroblocks alternatively too
E8_:	Encrypting half the I-Mb's and the headers of the non-intra macroblocks
E9_:	Encrypting 64-bit segment from alternative macroblocks regardless of the macroblock type

Also, it is recommended to read the ReadMe.txt files in each folder.

Nomenclature

MPEG	Moving Pictures Expert Group
I-MB	Intra-coded macroblock
GOP	Group of Pictures
τ	The ratio between the encrypted data and the total data in all frames of the MPEG video sequence
MIPS	one-Million-Instructions-Per-Second
DES	Private-key algorithm stands for Data Encryption Standard
RSA	Public-key algorithm stands for Rivest-Shamir-Adelman
C	Cost of one machine
L	Height of the frame in an MPEG video
W	Width of the frame in an MPEG video
t_{mb}	Time to decode one macroblock
t_{imb}	Time to decode one intra-coded macroblock
t_{fmb}	Time to decode one forward-predicted macroblock
t_{bmb}	Time to decode one backward-predicted macroblock
r_f	Ratio of the forward-predicted macroblocks in one frame
r_b	Ratio of the backward-predicted macroblocks in one frame
r_i	Ratio of the interpolated-predicted macroblocks in one frame
VLC	Variable Length Code

Bibliography

[1] Bernard Sklar, *Digital Communications: Fundamentals and Applications*. Englewood Cliffs, N.J.: Prentice Hall, 1988.

[2] Benoit M. Macq and Jean J. Quisquater, " Cryptography for Digital TV Broadcasting," *Proceedings of the IEEE*, vol. 83, N. 6, June 1995.

[3] Kh. Al-Jabri and A. Kh. Al-Asmari, "Secure Progressive Transmission of Compressed Images, " *IEEE transactions on consumer electronics*, vol. 42, N. 3, August 1996.

[4] T. M. Batchair, Y. Xu, and C. N. Zhang, "A secure Video on Demand systems," *Proceedings of the 1995 IEEE 14th annual international phoenix conference on computers and communications*.

[5] Iskander Agi and Li Gong, "An Empirical Study of Secure MPEG Video Transmissions," *Proceedings of SNDSS*, 1996.

[6] George A. Spanos and Tracy B. Maples, "Security for Real-Time MPEG Compressed Video in Distributed Multimedia Applications," *Proceedings of the 1996 IEEE 15th annual international phoenix conference on computers and communications*.

[7] Thomas Sikora, "MPEG Digital Video-Coding Standard," *IEEE signal processing magazine*, September 1997.

[8] Didier Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, Vol. 34, No. 4, April 1991.

[9] Frank S. Smith, "Encryption: What It Is and What It Can Do," *Journal of accounting and EDP*, Fall 1988.

[10]Bruce Schneier, *Applied Cryptography, Second Edition, Protocols, Algorithms and Sourse Codes in C.*. John Willey & Sons, Inc., 1996.

[11]C. E. Shannon, "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, July 1948, p.379; Oct. 1948, p.623.

[12]M. Christionsem and A. Hvidsten, "Cryptovision, a PAL Video Scrambler," *IEEE International Conference on Consumer Electronics*, 1988.

[13]K. R. Rao and J. J. Hwang, *Techniques and Standards for Image, Video and Audio Coding.* Prentice-Hall Inc., 1996.

[14]Roger J. Clarks, *Digital Compression of Still Images and Video.* Academic Press, 1996.

[15]Hewllet-Packard Company, *MPEG-2 Digital Video Technology and Testing.* 1995.

[16]William Sweet, "Chiariglione and the Birth of MPEG," *IEEE Spectrum*, September 1997.

[17]ISO/IEC 11172 Information Technology: Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/second, Part1: System; Part2: Video; Part3: Audio, 1993.

[18] YongCheng Li, Zhigang Chen, See-Mong Tan and Roy H. Campbell, "Security Enhanced MPEG Player," *Proceedings of International Workshop on Multimedia Software Development*, 1996.

[19] Thomas Sikora, "MPEG Digital Audio and Video Coding Standards," *IEEE Signal Processing Magazine*, September 1997.

[20] Leonardo Chiarigliene, "MPEG and Multimedia Communications," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 1, February 1997.

[21] Ralf Schäfer and Thomas Sikora, "Digital Video Coding Standards and Their Role in Video Communication," *Proceedings of the IEEE*, Vol. 83,No. 6June 1995.

[22] Leonardo Chiarigliene, "MPEG, A Technological Basis for Multimedia Applications," *IEEE Multimedia*, spring 1995.

[23] Ibrahim Al-Kadi, "Electronic Banking Security," *IEEE 4th Annual Exchange Meeting, KFUPM*, Dhahran, May 1997.

[24] Charles Blair, "Notes on Cryptogtraphy," Business Administration Dept., U. of Illinois, 1994.

[25] Haeng-Soo and Seung-Jo Han, "Design of the Extended-DES Cryptography," *Proceedings of 1995 IEEE International Symposium on Information Theory.*

[26] Bruce Schneier, "Cryptography, Security, and the Futures," *Communications of the ACM*, Vol. 40, No. 1, January 1997.

[27]Stene Steinberg, "A student's View of Cryptography in Computer Science," *Communications of the ACM*, Vol. 34, No. 2, February 1991.

[28]G. A. Spanos and T. B. Maples, "Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video," *Proceedings of the 4th International Conference on Computer Communications and Networking*, Las Vegas, Nevada. October 1995.

[29]T. Kunkelmann and R. Reinema, "A Scalable Security Architecture for Multimedia Communication Standard," *IEEE International Conference on Multimedia Computing and system*, 1997.

[30]J. Fery, K. T. Lo and H. Mehrpour, "*Scene Change Detection Algorithm for MPEG Video Sequence*," Proceedings of International Conference on Image Processing, 1996.

[31]W. Fumy and P. Landrock, "Principles of Key Management," *IEEE Journal on Selected Areas in Communications*. Vol. 11, No. 5, June 1993.

[32]E. Biham and A. Shamir, "Differential Cryptanalysis of the DES," Springer-Verlag, 1993.

[33]C. Boyd, "Modern Data Encryption," *Electronics and Communication Engineering Journal*, October 1993.

[34]E. F. Brickell and A. M. Odlyzko, "Cryptanalysis: A Summary of Recent Results," *Proceedings of the IEEE*, Vol. 76, No. 5, May 1988.

[35]J. L. Massey, "An Introduction to Contemporary Cryptology," *Proceedings of the IEEE*, Vol. 76, No. 5, May 1988.

[36]T. Denny, B. Dodson, A. K. Lenstra and N. S. Manasse, "On the Factorization of RSA-120," *Advances in Cryptology-CRYPTO'93 Proceedings*, Springer-Verlag, 1994, pp. 166-174.

[37]A. K. Lenstra and M. S. Manasse, "Factoring with Two Large Primes," *Advances in Cryptology-CRYPTO'90 Proceedings*, Springer-Verlag 1991, pp. 72-82.

[38]A. K. Lenstra and M. S. Manasse, "Factoring by Electronic Mail," *Advances in Cryptology-EUROCRYPT'89 Proceedings*, Springer-Verlag 1990, pp. 355-371.

[39]M. J. B. Robshaw, "Security Estimates for 512-bit RSA," *Proceedings of WESCON'95*, November 1995.

[40]A. M. Odlyzko, "The Future of Integer Factorization," *CryptoBytes*, Vol. 1, No. 2, Summer 1995.

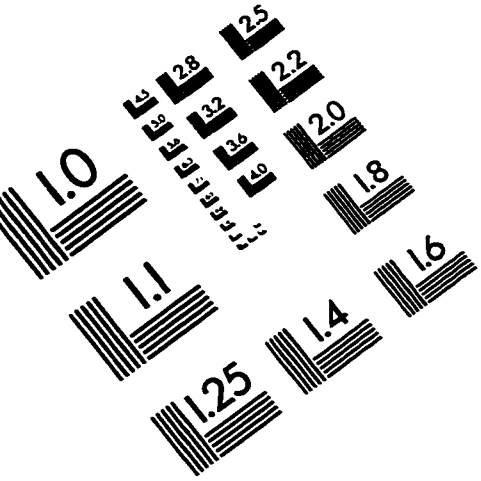
[41]Didier J. LE Gall, " The MPEG video compression algorithm," *Signal Processing: Image Communication* 4 (1992) 129-140 Elsevier.

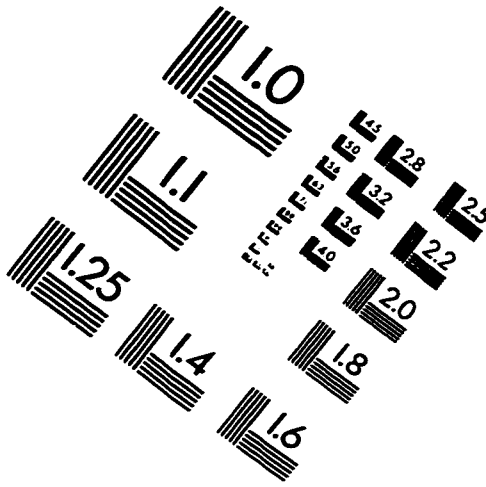
[42]Joan Mitchell, William Pennebaker, Chad Fog, and Didier LeGall, *MPEG video compression standard.* Chapman & Hall, 1996.

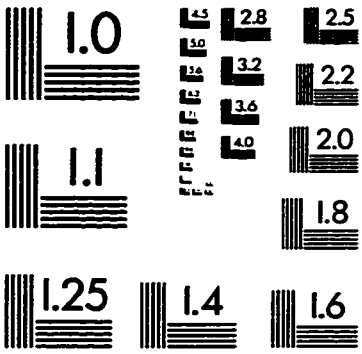
[43]Raj Talluri, "Error-Resilient Video Coding in the ISO MPEG-4 standard," *IEEE Communications Magazine*, June 1998.

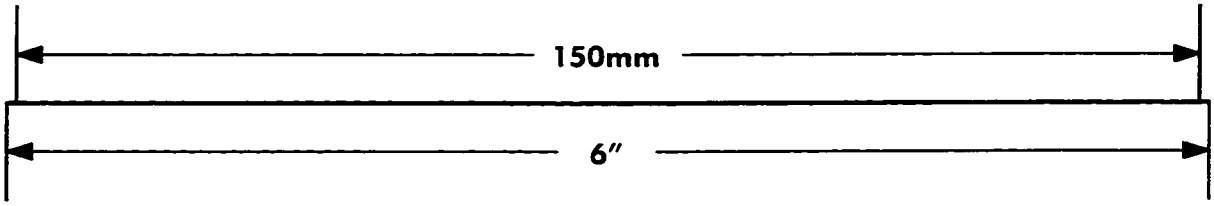
[44]A. Puri and A. Eleftheriadis, " MPEG-4: An Object-based Multimedia Coding Standard for Supporting Mobile Applications," *ACM Mobile Networks and Apps. J.*, 1998.

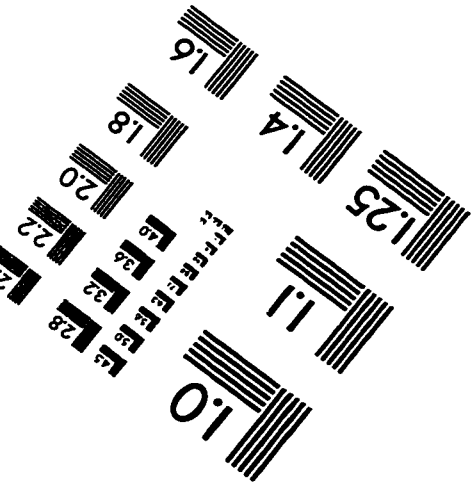
IMAGE EVALUATION
TEST TARGET (QA-3)

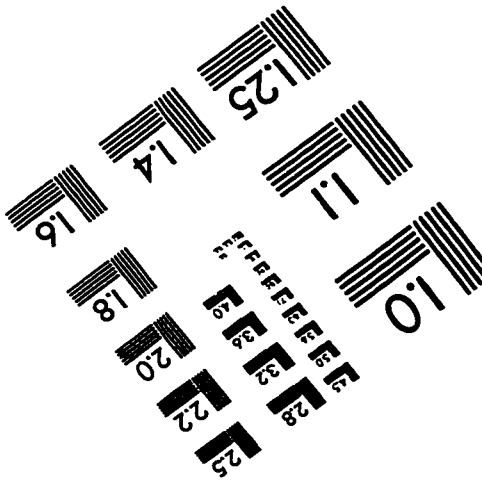












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989